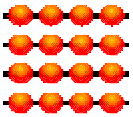
UNIVERSIDADE FEDERAL DO CEARÁ
DEPARTAMENTO DE COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO



Dissertação de Mestrado

Atualização de Múltiplas Bases de Dados através de Mediadores

por

Bernadette Farias Lóscio

Orientadora: Profa. Dra. Vânia Maria Ponte Vidal

Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Mestrado em Ciência da Computação

Bernadette Farias Lóscio

Atualização de Múltiplas Bases de Dados através de Mediadores

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará como requisito parcial para obtenção de grau de Mestre em Ciência da Computação.

Orientadora: Profa. Dra. Vânia Maria Ponte Vidal

AGRADECIMENTOS

Agradeço a Deus,
Agradeço aos meus pais Roberto e Roceli,
Agradeço as minhas irmãs Paula e Cláudia e ao meu irmão Roberto, meus grandes amigos e incentivadores
Agradeço ao Luciano pela paciência nas horas em que não pude estar com ele e por todas as horas em que ele esteve comigo
Agradeço a minha orientadora Vânia por toda a paciência e todos os ensinamentos que enriqueceram este trabalho e que com certeza
Agradeço a Roberta, minha primeira orientadora,
Agradeço aos meus amigos
Agradeço aos professores
Agradeço a Lia

RESUMO

Muitas aplicações atuais requerem o acesso integrado a informações armazenadas em várias fontes de informação com diferentes modelos de dados e mecanismos de acesso. A arquitetura de mediadores tem sido adotada por muitos projetos que buscam solucionar esse problema de prover acesso integrado a múltiplas fontes de informação. Nessa arquitetura, os mediadores são "interfaces" através das quais os usuários consultam e atualizam múltiplas bases de dados.

Nesta dissertação abordamos o problema de Atualização de Múltiplas Bases de Dados através de Mediadores. Uma atualização de mediador é simplesmente uma atualização que é especificada em um mediador, mas que deve ser traduzida em uma seqüência de atualizações nos bancos de dados locais. O problema de *Tradução de Atualização de Mediadores* (TAM) refere-se à questão de definir traduções corretas para as atualizações submetidas através de mediadores.

Neste trabalho desenvolvemos algoritmos para gerar tradutores para as operações básicas de atualização de mediadores. Um tradutor é uma função que recebe como entrada um pedido de atualização de mediador e gera a tradução para esta atualização. Os tradutores são definidos em tempo de projeto e armazenados juntamente com a definição do mediador. Assim, uma vez definido o tradutor, o usuário especifica atualizações através do mediador e o tradutor as traduz em atualizações nos bancos de dados locais, sem a necessidade de qualquer diálogo adicional.

Nos algoritmos propostos, os tradutores são gerados a partir das assertivas de correspondência do mediador, que especificam formalmente o relacionamento do esquema do mediador com os esquemas locais. O uso de assertivas de correspondência, nos permite provar formalmente que os tradutores gerados pelos nossos algoritmos produzem traduções corretas. É importante notar que os algoritmos propostos também podem ser utilizados para a definição dos tradutores de atualizações de visões em um sistema de banco de dados

centralizado. Visões, assim como os mediadores, constituem interfaces através das quais os usuários consultam e atualizam um banco de dados.

ABSTRACT

Many current applications require integrated access to information stored in several information sources with different data models and access mechanism. The mediator architecture has been adopted in many projects to provide integrated access to multiple information sources that can be autonomous and heterogeneous. Mediator is a facility that supports an integrated view over multiple information sources, and allows for queries to be made against the integrated view.

In this work, we extend the mediator architecture to support updates against the integrated view. Updates expressed against the mediator's integrated view need to be translated into updates of the underlying local databases. The problem of defining *Mediator Update Translations* refers to the question of defining correct translations for the updates expressed against the mediator's integrated view.

In this work we developed algorithms to generate translators for the basic types of mediator update operations. In our approach, a translator is a function that receives an update request and generates the update's translation. The translators are defined at mediator definition time, and their definitions are stored along with the mediator's specification. Thus, once defined the translators, the user can express updates against the mediator and the translator generates the update's translation.

In our algorithms, the translators are generated based on the correspondence assertions that formally specify the relationships between the mediator schema and the local schemas. The use of correspondence assertions allows us to formally prove that the translators generated by our algorithms produce correct translations. Our approach can also be applied to the definition of view update translators in centralized databases.

ÍNDICE

LISTA DE FIGURAS

CAPÍTULO 1

INTRODUÇÃO

A evolução dos sistemas de informação delineia-se pela utilização de aplicações que envolvem o acesso e a manipulação de dados a partir de muitos bancos de dados preexistentes . É a chamada tendência da descentralização, que deixa de lado o princípio onde aplicações devem trabalhar isoladamente e passa a adotar a cooperação intersistemas.

Para impulsionar ainda mais essa tendência, nas últimas décadas, temos testemunhado uma explosão espetacular na quantidade de dados disponíveis em meio eletrônico. Porém, esta vasta quantidade de dados têm sido organizada e armazenada por pequenos grupos de usuários, trabalhando para diferentes organizações, em problemas diversos. Consequentemente, os sistemas têm sido desenvolvidos usando diferentes especificações e modelos de dados, sendo assim imperativo encontrar meios para contornar estas diferenças no sentido de que os sistemas possam trabalhar cooperativamente.

Além da grande disponibilidade de dados, os rápidos avanços tecnológicos na comunicação de dados, permitindo a implementação de redes de alta velocidade usando fibras óticas, têm contribuído para aumentar o interesse nas aplicações que acessam múltiplas bases de dados. Entretanto, estas melhorias devem ser complementadas com avanços de software para obtermos um benefício real, senão ganharemos acesso a mais dados, mas não melhoraremos o acesso e a qualidade das informações necessárias para a tomada de decisão.

Assim, dada a evidência da disponibilidade de dados e das mudanças na capacidade física dos sistemas de informação, o próximo passo é criar um ambiente que permita o compartilhamento e a troca de informações entre bancos de dados autônomos e heterogêneos. Uma proposta para solucionar o problema de integração de múltiplas bases de dados é o uso de mediadores [Wiederhold92]. Nessa proposta, o acesso e a atualização de dados distribuídos em múltiplas bases de dados é efetuado através de consultas e atualizações submetidas ao sistema através de mediadores, que são os responsáveis por transformá-las em subconsultas e atualizações a serem executadas nas bases de dados correspondentes. Cada mediador encapsula informações sobre as bases de dados para as quais ele foi construído, possibilitando que eles sejam desenvolvidos independentemente e depois sejam agrupados, formando “redes de mediadores”.

O uso de mediadores tem se mostrado adequado para os futuros sistemas de informação, caracterizados pela integração em grande escala de múltiplas bases de dados. Mediadores são módulos de software capazes de executar consultas e atualizações sobre informações distribuídas em múltiplas bases de dados. Cada mediador representa a visão que o usuário tem das bases de dados locais para as quais ele foi construído e funciona como uma interface entre o usuário e estas bases de dados.

Construir módulos de software, como os mediadores, não é uma tarefa fácil, pois eles devem oferecer serviços que possibilitem a integração e o refinamento de dados distribuídos em diferentes bases de dados. Uma das dificuldades a ser considerada é a heterogeneidade na representação de um mesmo conceito do mundo real, que pode estar armazenado em diferentes fontes de informação. O mediador deve fornecer uma visão integrada e consistente dos dados, independente da localização e representação dos mesmos. Além disso, é necessário estabelecer correspondências entre os dados do mediador e os dados das bases de dados locais, para que possamos definir regras de mapeamento entre o estado do mediador e os estados dos bancos de dados locais. Esse mapeamento é necessário para permitir que dados locais sejam consultados e atualizados através dos mediadores.

Tendo em vista a complexidade encontrada na construção de mediadores, propomos em [Vidal97] uma metodologia para projeto de mediadores consistindo dos seguintes passos:

(i) *Especificação do esquema do mediador*: Durante esse passo, os requisitos dos usuários do mediador são analisados e o esquema do mediador é especificado usando um modelo de dados de alto nível;

(ii) *Integração do esquema do mediador*: O objetivo desse passo é fazer a integração do esquema do mediador com os esquemas locais para identificar as assertivas de correspondência que especificam formalmente o relacionamento do esquema do mediador com os esquemas locais;

(iii) *Especificação do Mediador*: A partir do esquema do mediador e das assertivas de correspondência, define-se o mapeador de estados e o mapeador de atualizações do mediador.

Neste trabalho, abordamos a fase de *Especificação do Mediador* relacionada a etapa de definição do mapeador de atualizações da metodologia proposta em [Vidal97]. O mapeador de atualizações especifica como atualizações requisitadas através do mediador são traduzidas em atualizações especificadas nos bancos de dados locais.

A atualização de múltiplas bases de dados através de mediadores é um problema que ainda não foi tratado na literatura. As pesquisas atuais têm focalizado o uso de mediadores apenas para o acesso integrado a informações distribuídas em múltiplas bases de dados. O problema de atualização de múltiplas bases de dados através de mediadores é muito semelhante ao problema de atualização de bancos de dados através de visões, pois assim como os mediadores, visões constituem as interfaces através das quais os usuários consultam e atualizam o banco de dados. Dessa forma, atualizações submetidas a um banco de dados, tanto através de um mediador como de uma visão, devem ser traduzidas em atualizações a serem executadas nas bases de dados correspondentes.

Como o problema de atualização de múltiplas bases de dados através de mediadores é muito semelhante ao problema de atualização de bancos de dados através de visões, alguns resultados obtidos nas pesquisas desenvolvidas para tratar a atualização através de visões podem ser aplicados para a atualização através de mediadores. Entre os enfoques propostos para a atualização através de visões podemos destacar duas abordagens. A primeira trata as visões como tipos de dados abstratos onde os tradutores são armazenados juntamente com a definição da visão. A segunda abordagem consiste em definir procedimentos gerais de tradução [Dayal82, Keller86a, Bancilhon81, Larson91]. Estes procedimentos recebem como entrada a definição da visão, uma atualização de visão e o estado atual do esquema. Eles produzem, se possível, uma tradução da atualização de visão em atualizações no esquema conceitual, que satisfaçam algumas propriedades.

Apesar do grande número de pesquisas que tratam a atualização de bancos de dados através de visões, este problema ainda não foi resolvido satisfatoriamente. Isto acontece porque a maioria dos enfoques propostos não usa um formalismo para representar os relacionamentos entre o esquema da visão e o esquema do banco de dados. Com essa falta de formalismo não é possível garantir que as traduções geradas pelos algoritmos propostos nesses enfoques são corretas.

Neste trabalho, desenvolvemos algoritmos que geram tradutores para as operações básicas de atualização de mediadores. Nós propomos que seja gerado um tradutor para cada uma das operações de atualização de mediador que forem permitidas. Depois de gerados, os tradutores são armazenados juntamente com a definição do mediador.

Nos algoritmos propostos, os tradutores são gerados a partir das assertivas de correspondência do mediador, que especificam formalmente o relacionamento do esquema do mediador com os esquemas locais. A vantagem do uso de assertivas de correspondência, para especificar formalmente os relacionamentos entre o esquema do mediador e os esquemas dos BDs locais, é que nos permite provar formalmente que os tradutores gerados pelos nossos algoritmos produzem traduções corretas.

Uma das contribuições deste trabalho é a extensão do modelo de Entidades e Relacionamentos com o formalismo necessário para representar as várias formas de correspondências entre esquemas (assertivas de correspondência). Uma das limitações das metodologias de integração de esquemas existentes [Navathe et al.86, Spaccapietra94] é que não se pode representar a correspondência de conceitos semelhantes representados de formas diferentes. Neste trabalho, propomos formas gerais de assertivas de dependência existencial que permitem especificar formalmente os tipos mais comuns de correspondência de conceitos com múltiplas representações.

É importante notar que os algoritmos desenvolvidos neste trabalho para tratar o problema de atualização através de mediadores também se aplicam para o problema de atualização através de visões.

A dissertação é divida em nove capítulos como se segue. No capítulo 2, apresentamos as principais características e componentes da arquitetura de mediadores. No capítulo 3, apresentamos o modelo ER Estendido, que utilizamos para representar o esquema do mediador e os esquemas locais. No capítulo 4, descrevemos mais detalhadamente a metodologia para projeto de mediadores proposta em [Vidal97]. No capítulo 5, apresentamos o nosso enfoque para atualização de múltiplas bases de dados através de mediadores e discutimos alguns trabalhos relacionados. No capítulo 6, descrevemos o algoritmo que gera tradutores para as operações de modificação de atributos de mediador. No capítulo 7, descrevemos os algoritmos que geram tradutores para as operações de adição e remoção em tipos de entidade de mediador. No capítulo 8, descrevemos os algoritmos que geram tradutores para as operações de adição e remoção em tipos de relacionamento de mediador. No capítulo 9, apresentamos as nossas conclusões e trabalhos futuros.

CAPÍTULO 2

ARQUITETURA DE MEDIADORES

Neste capítulo abordaremos a arquitetura de mediadores, que tem sido usada por muitos projetos para fornecer acesso integrado a múltiplas bases de dados. Na seção 2.1 apresentamos as principais características e o suporte necessário para esta arquitetura. Na seção 2.2 apresentamos os mediadores, que funcionam como interfaces através das quais os usuários podem acessar e consultar, de forma transparente, múltiplas bases de dados. Na seção 2.3 apresentamos alguns projetos que atualmente têm utilizado a arquitetura de mediadores e na seção 2.4 descrevemos outras arquiteturas utilizadas para prover acesso integrado a múltiplas bases de dados.

2.1 Características Gerais da Arquitetura de Mediadores

Muitas aplicações atuais requerem o acesso integrado a informações armazenadas em várias fontes de informação com diferentes modelos de dados e mecanismos de acesso. A arquitetura de mediadores tem sido usada por muitos projetos [Wiederhold92, Chawathe et al.94] para prover acesso integrado a múltiplas fontes de informação. Os componentes da arquitetura de mediadores são apresentados na *Figura 2.1* e descritos a seguir:

– *Fontes de Informação:* podem ser autônomas e heterogêneas, porque na maioria das vezes elas não foram especificadas com a intenção de compartilhamento. As fontes de informação (FIs) podem ser bancos de dados, depósitos de objetos, bases de conhecimento, bibliotecas digitais e até mesmo sistemas de recuperação de informação.

– *Tradutores:* convertem os dados das FIs para um modelo de dados comum e convertem consultas de aplicações em consultas específicas da fonte de informação correspondente.

– *Mediadores*: são "interfaces" através das quais os usuários consultam e atualizam múltiplas bases de dados.

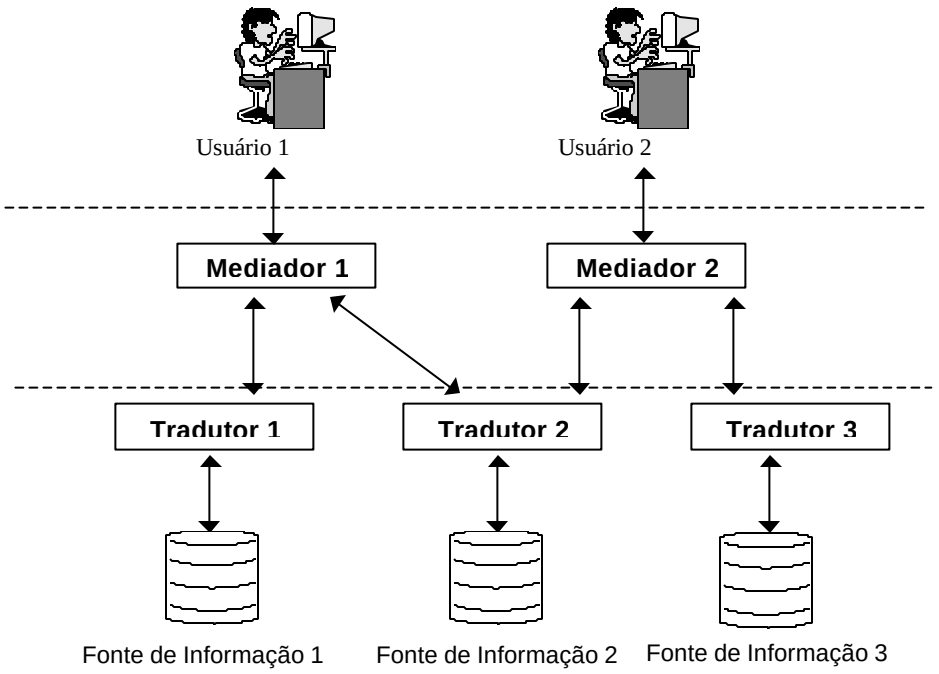


Figura 2.1 – Arquitetura de Mediadores

Na arquitetura de mediadores o acesso aos dados distribuídos em múltiplas fontes de informação é efetuado através de consultas que são submetidas ao sistema através do mediador, e este as transforma em subconsultas a serem enviadas às fontes de informação.

As subconsultas geradas pelo mediador devem ser traduzidas para linguagens de consulta de cada SGBD componente. Ao final, os resultados das consultas são traduzidos e a resposta é retornada para o usuário. O processo de atualização dos dados através de mediadores é semelhante ao de consulta; as atualizações submetidas ao sistema, via mediador, devem ser traduzidas em atualizações a serem executadas nas bases de dados correspondentes.

Para dar suporte à arquitetura de mediadores são necessários:

- *Modelo de Dados Comum*

Um dos grandes problemas com a integração de informações em múltiplas bases de dados é a heterogeneidade na representação dessas informações, que podem estar armazenadas em bancos de dados relacionais, mas também podem estar armazenadas em repositórios de objetos, bases de conhecimento e sistemas de arquivos.

Os componentes da arquitetura de mediadores precisam trocar informações, ou como resposta para o usuário final ou para integração com outros componentes. Deve existir um acordo entre os componentes de como as informações serão requisitadas, como elas serão representadas e como serão transportadas ao longo de uma rede. Para isso é necessário que os componentes utilizem um modelo de dados comum. Esse modelo deve ser flexível, oferecer uma coleção rica de estruturas e fornecer informações sobre as estruturas (meta-informações). Como exemplo de modelo de dados comum podemos citar o OEM – Object Exchange Model adotado pelo projeto TSIMMIS [Chawathe et al.94].

- *Linguagem de Consultas Comum*

Da mesma maneira que existe a necessidade de um modelo de dados comum para a troca de informações entre os componentes de um sistema, existe a necessidade de uma linguagem de consultas comum para permitir a comunicação entre os componentes. A adoção de uma linguagem de consultas comum permite que novos mediadores e novas fontes de informação se juntem a mediadores e fontes já existentes.

❖ *Ferramentas para Geração de Tradutores e Mediadores*

Implementar mediadores e tradutores pode ser uma tarefa complicada e que consome muito tempo. Porém, uma boa parte do trabalho envolvido na codificação de tradutores e mediadores pode ser automatizada. Sendo assim, devem existir ferramentas que possam gerar mediadores e tradutores automaticamente ou semi-automaticamente a partir de uma especificação de alto nível.

2.2 Mediadores

Os *mediadores* são módulos de software que exploram o conhecimento representado em um conjunto ou subconjunto de dados para gerar informações para aplicações residentes em uma camada superior [Wiederhold92].

Um mediador oferece uma *visão integrada* das informações distribuídas em múltiplas bases de dados e fornece informações mais úteis aos usuários. Para isso ele deve reestruturar informações e fornecer abstrações dessas informações. Além do acesso, um mediador também deve suportar atualizações sobre as informações distribuídas em múltiplas bases de dados.

Tipicamente, um mediador é criado para um dado domínio de interesse associado a fontes de informação específicas ou subconjuntos dessas fontes. Uma aplicação que depende de múltiplos domínios usará múltiplos mediadores. Em alguns casos, as fontes de informação usadas pelos mediadores serão atômicas, isto é, bancos de dados auto-contidos, bases de conhecimento, dicionários, etc. Em outros casos, mediadores podem acessar fontes de informação não-atômicas, tais como outros mediadores. Para um melhor entendimento dos mediadores, a seguir listamos algumas de suas características [Wiederhold94]:

❖ *Domínio Específico*

Mediadores são especializados, ou seja, são construídos para domínios específicos. A especialização torna a manutenção possível, visto que um especialista em um determinado domínio não precisa considerar restrições impostas pela manipulação de domínios não relacionados. Além disso, diferentes domínios possuem características distintas e podem requerer diferentes estruturas para representar seus dados, ou até mesmo podem ser melhor servidos por diferentes paradigmas de programação.

❖ *Distribuição*

Um mediador integra e filtra dados disponíveis em uma rede com o objetivo de torná-los mais úteis e relevantes. Esta função pode ser alcançada no computador onde o mediador foi desenvolvido ou pode ser distribuída para outros computadores da rede. No caso em que a demanda por um determinado mediador é alta, réplicas podem ser distribuídas em *sites* adicionais da rede, pois à medida em que os links de comunicação diminuem, o tempo de resposta será melhorado.

❖ *Manutenção*

A mediação enriquece dados disponíveis em uma rede de computadores aplicando o conhecimento do especialista que cria o mediador. Mediadores também devem ser mantidos por esse especialista, para que permaneçam úteis em um mundo constantemente em mudanças. Mediadores pobres em manutenção perderão seu valor ao longo do tempo e são fortes candidatos à troca por um mediador mais atual.

❖ *Compartilhamento*

Os mediadores serão mais úteis se puderem servir a uma variedade de aplicações. As aplicações, por sua vez, irão compor ao máximo suas tarefas, adquirindo informações a partir de um conjunto disponível de mediadores. Informações não disponíveis podem motivar a criação de novos mediadores.

2.3 Projetos que usam Mediadores

Dentre os projetos que atualmente utilizam mediadores para prover acesso integrado a múltiplas bases de dados podemos destacar o TSIMMIS [Chawathe et al.94] e o HERMES [Subrahmanian et al.95]. O TSIMMIS é um projeto da Universidade de Stanford e IBM Alamaden Research Center e o HERMES é um projeto desenvolvido na Universidade de Maryland.

O objetivo do projeto TSIMMIS é desenvolver ferramentas que facilitem a rápida integração de fontes de informação heterogêneas que podem incluir tanto dados estruturados como não-estruturados.

O TSIMMIS adota o modelo de dados *OEM (Object Exchange Model)* que permite a troca de informações via objetos auto-descritivos entre diferentes tipos de fontes de informação. Como linguagem de consulta, o TSIMMIS adota o *OEM-QL* e como linguagem de especificação do mediador adota *MSL (Mediator Specification Language)* que é uma linguagem de alto nível, baseada em regras.

O projeto TSIMMIS não tem como objetivo executar uma integração de informações completamente automatizada que oculta todas as diversidades dos usuários, mas ao invés disso fornecer ferramentas para auxiliar as atividades de integração e processamento de informações.

O projeto HERMES propõe um sistema para integração de múltiplas fontes de informações, que apresenta facilidades para a construção de mediadores. A motivação principal por trás do HERMES é modularizar as atividades envolvidas na criação do mediador. O HERMES fornece um conjunto de ferramentas para auxiliar o autor do mediador na construção de mediadores. Algumas ferramentas são úteis para a construção do código do mediador e outras são úteis para a construção do código de acesso às fontes de dados. Essas ferramentas juntamente com a linguagem de especificação do mediador

formam o ambiente de programação do mediador. A linguagem adotada por esse projeto para especificar o mediador é uma linguagem de alto nível, declarativa e baseada em regras.

Atualmente, sistemas que utilizam a arquitetura de mediadores para integração de múltiplas bases de dados preocupam-se em oferecer ferramentas que auxiliem a construção de mediadores. Busca-se, principalmente, a geração automática de mediadores a partir de uma declaração de alto nível dos requisitos do mediador. O TSIMMIS e o HERMES oferecem geradores automáticos e um ambiente de programação (composto por várias ferramentas), respectivamente, para auxiliar a construção de mediadores.

2.4 Outras Arquiteturas para Integração de Múltiplas Bases de Dados

Além do uso de mediadores para fornecer acesso integrado a informações distribuídas em múltiplas bases de dados, podemos destacar algumas soluções clássicas para esse problema. Entre estas soluções destacamos os *sistemas de bancos de dados não-federados*, os *sistemas de bancos de dados federados fortemente* e *fracamente acoplados*. Esses sistemas pertencem a uma mesma categoria denominada de *sistemas de múltiplos bancos de dados*. A classificação acima é estabelecida de acordo com a autonomia dos bancos de dados componentes, como mostrado nas definições abaixo:

- *Sistema de banco de dados não federado*: nesse caso os sistemas de bancos de dados componentes não são autônomos. Existe apenas um nível de gerenciamento, onde operações locais e globais são executadas igualmente.

- *Sistema de banco de dados federado*: é uma coleção de sistemas de bancos de dados cooperantes e autônomos que participam da federação para permitir um compartilhamento parcial e controlado de seus dados. Nesse caso existe uma distinção entre usuários locais e globais. A característica chave de uma federação é a cooperação entre sistemas independentes.

A arquitetura de cinco níveis de um sistema de banco de dados federado [Sheth90] é apresentada na *Figura 2.2* e seus componentes são:

1. *Esquema local* é o esquema conceitual do sistema de banco de dados componente.
2. *Esquema componente* é derivado do esquema local pela sua tradução para um modelo de dados comum.
3. *Esquema exportado* representa o subconjunto de dados do esquema componente que estão disponíveis para a federação.
4. *Esquema federado* é a integração de múltiplos esquemas exportados.
5. *Esquema externo* define um esquema para um usuário, aplicação ou classes de usuários e aplicações.

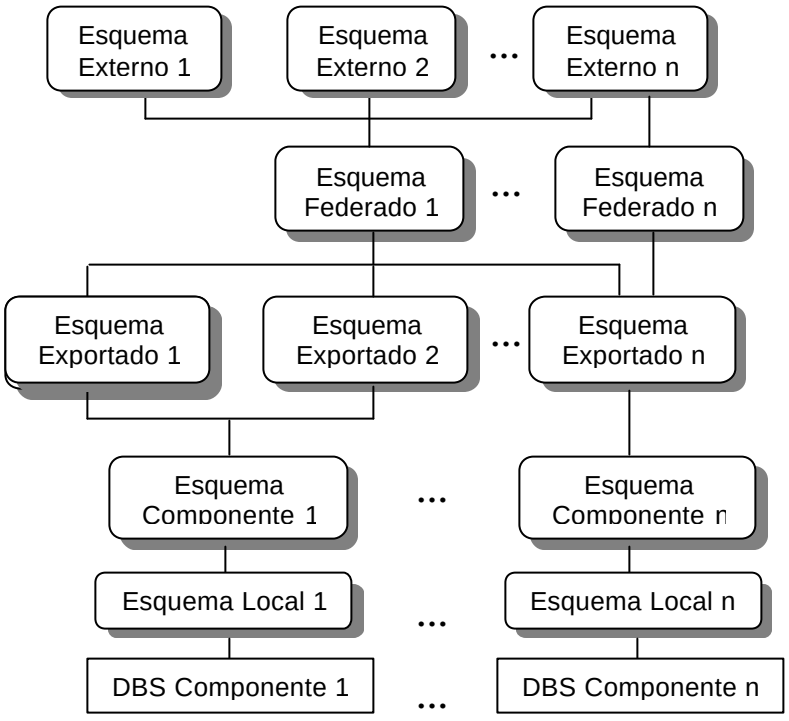


Figura 2.2 – Arquitetura de cinco níveis de um Sistema de Banco de Dados Federado

Os esquemas exportados são definidos a partir de negociações entre o DBA da federação e o DBA do sistema de banco de dados componente que tem controle sobre o que está incluído no esquema exportado. Os esquemas federados são responsabilidade do DBA da federação que deve criá-los e controlá-los. Os esquemas externos são criados a partir da negociação entre o usuário da federação e o DBA da federação que tem autoridade sobre o que está incluído em cada esquema externo.

Diferentes arquiteturas de sistemas federados são criadas de acordo com o nível de integração dos bancos de dados componentes e com o responsável pelo gerenciamento da federação. Um sistema federado possui duas categorias clássicas: *fracamente* e *fortemente* acoplado.

❖ *Sistemas de Bancos de Dados Federados Fracamente Acoplado*

Um sistema de banco de dados federado é fracamente acoplado se for responsabilidade do usuário criar e manter a federação e não existir controle por parte da federação ou de seus administradores. Cada usuário da federação é responsável por criar e administrar o seu esquema federado. Definir um esquema federado fracamente acoplado é como definir uma visão sobre os esquemas dos bancos de dados componentes. O usuário examina os esquemas exportados para determinar quais aqueles que descrevem dados que ele gostaria de acessar e, em seguida, ele define o esquema federado a partir da importação de objetos dos esquemas exportados. Isso pode ser feito através de uma interface com o usuário, uma aplicação ou definindo uma consulta em uma linguagem apropriada que faz referência a objetos dos esquemas exportados. É responsabilidade do usuário entender a semântica dos objetos nos esquemas exportados e resolver as heterogeneidades semânticas. Nesse enfoque, o esquema federado pode ser acessado ou excluído a qualquer momento pelo usuário da federação.

❖ *Sistemas de Bancos de Dados Federados Fortemente Acoplado*

Uma federação é fortemente acoplada quando existe uma autoridade central responsável por criar e manter a federação e ativamente controlar o acesso aos bancos de dados componentes. Um sistema federado fortemente acoplado pode suportar uma ou várias federações. No primeiro caso existe apenas um esquema federado e todos os acessos são feitos através dele. Como o esquema federado é criado pela integração de todos os esquemas exportados e como ele deve suportar os requisitos de todos os usuários da federação, este esquema federado pode se tornar muito grande e difícil de criar e manter. No segundo caso existem vários esquemas federados, que são definidos de acordo com classes de usuários da federação com diferentes requisitos para acesso aos dados.

Neste capítulo apresentamos algumas arquiteturas utilizadas para prover acesso integrado a informações armazenadas em múltiplas bases de dados. Foram apresentadas a arquitetura de mediadores, os sistemas de bancos de dados não-federados e os sistemas de bancos de dados federados fortemente e fracamente acoplados. Dentre as arquiteturas apresentadas, a arquitetura de mediadores tem se mostrado a mais adequada para os futuros sistemas de informação, caracterizados pela integração em grande escala de múltiplas bases de dados.

CAPÍTULO 3

MODELO ER ESTENDIDO

Neste trabalho adotamos o modelo de Entidades e Relacionamentos Estendido, para representar o esquema do mediador e os esquemas dos bancos de dados locais. Esse modelo é resultado da extensão do modelo ER com o formalismo necessário para representar as várias formas de correspondências entre esquemas. Uma das vantagens desse modelo é que ele permite especificar formalmente os tipos mais comuns de correspondências de conceitos com múltiplas representações. Na seção 3.1 deste capítulo apresentamos os conceitos preliminares do modelo de Entidades e Relacionamentos Estendido e na seção 3.2 definimos os vários tipos de assertivas de correspondência que são suportadas pelo modelo.

3.1 Conceitos Preliminares

Após mais de vinte anos de uso, o modelo ER continua sendo o modelo mais usado para a modelagem conceitual dos dados. Esta popularidade se deve aos seguintes fatores: (i) é um modelo relativamente fácil de usar, (ii) a existência de uma grande variedade de ferramentas CASE que suportam este modelo, (iii) muitas pessoas acreditam que entidades

e relacionamentos são conceitos de modelagem natural no mundo real. O modelo ER é rico em semântica, o que facilita o entendimento das correspondências existentes entre os componentes dos esquemas. Os conceitos básicos do modelo ER são entidades, relacionamentos e atributos.

Uma entidade é um objeto representado em nossa mente e que pode ser distintamente identificado. Entidades podem ser classificadas em diferentes tipos. Cada tipo de entidade contém um conjunto de entidades que satisfazem um conjunto de propriedades comuns predefinidas, por exemplo, o conjunto de empregados de uma companhia pode ser representado pelo tipo de entidade *Empregado*.

Em muitos casos, as entidades que compõem um tipo de entidade podem ser agrupadas em subgrupos adicionais. Por exemplo, entidades que são membros do tipo de entidade *Empregado* podem ainda ser agrupadas em secretário, engenheiro, etc. Os subgrupos são chamados de subclasses do tipo de entidade *Empregado* e *Empregado* é chamado de superclasse para cada uma destas subclasses. O relacionamento entre uma superclasse e qualquer uma de suas subclasses é chamado de relacionamento IS-A.

Um tipo de relacionamento é uma associação de tipos de entidades. Cada tipo de relacionamento $\boldsymbol{R}$ entre tipos de entidades $\boldsymbol{E_1,\ldots,E_n}$ define um conjunto de associações entre entidades de $\boldsymbol{E_1,\ldots,E_n}$. Nesse caso, dizemos que os tipos $\boldsymbol{E_1,\ldots,E_n}$ participam do tipo de relacionamento $\boldsymbol{R}$. Um relacionamento $\boldsymbol{r}$ de $\boldsymbol{R}$ é uma associação de entidades, onde cada associação inclui exatamente uma entidade de cada tipo de entidade participante de $\boldsymbol{R}$. O grau de um tipo de relacionamento é o número de tipos de entidades participantes.

Tipos de entidade e tipos de relacionamento possuem atributos que representam propriedades estruturais. Os atributos podem ser monovalorados ou multivalorados. Cada atributo $\boldsymbol{A}$ de um tipo de entidade ou tipo de relacionamento está associado com um domínio, $\boldsymbol{Dom}(\boldsymbol{A})$, que especifica os valores que podem ser atribuídos a $\boldsymbol{A}$. O número mínimo de atributos de um tipo de entidade cujos valores identificam uma única entidade deste tipo é chamado chave ou identificador. O identificador de um tipo de relacionamento

pode ser obtido do conjunto de identificadores de alguns de seus tipos de entidades participantes, ou da junção destes identificadores com atributos próprios do tipo de relacionamento. A definição formal de identificador será apresentada no final desta seção.

Neste trabalho, entidades e relacionamentos são especificados como se segue:

- A especificação de uma entidade ***e*** de um tipo ***E*** é dada por:
  $< E\{ <A_1 : v_1> <A_2 : v_2>\ldots<A_n : v_n>\}>$, onde $A_1 \ldots A_n$ são atributos de ***E*** e $v_1,\ldots,v_n$ são os valores destes atributos.

- A especificação de um relacionamento ***r*** de um tipo ***R*** é dada por:
  $<R \{<E_1 : e_1> <E_2 : e_2>\ldots<E_n : e_n> <A_1 : v_1> <A_2 : v_2>\ldots<A_k : v_k>\}>$, onde $E_1,\ldots,E_n$ são tipos de entidades participando de ***R***, $e_1,\ldots,e_n$ são entidades dos tipos $E_1,\ldots,E_n$, $A_1,\ldots,A_n$ são atributos de ***R*** e $v_1,\ldots,v_n$ são os valores destes atributos.

Um esquema ER é uma tripla $S = (\mathcal{E}, \mathcal{R}, \mathcal{I})$ onde $\mathcal{E}$ é um conjunto de tipos de entidade, $\mathcal{R}$ é um conjunto de tipos de relacionamento e $\mathcal{I}$ é um conjunto de restrições de integridade.

A seguir, definimos os conceitos de *Ligação*, *Caminho* e *Estado de Esquema*, os quais são necessários para a definição formal dos vários tipos de assertivas de correspondência que são apresentados na próxima seção.

Definição 3.1 (*Ligação*) : Sejam X_1 e X_2 elementos de um esquema (um elemento pode ser um tipo de entidade, tipo de relacionamento ou atributo), X_1-X_2 é uma ligação se:
(i) X_1 é um atributo de X_2 (ou vice-versa); X_1-X_2 é chamada uma *ligação de atributo*;
(ii) X_1 é um tipo de entidade participando de um tipo de relacionamento X_2 (ou vice-versa); X_1-X_2 é chamada uma *ligação de relacionamento*;□

Cada ligação de relacionamento X_1-X_2 está associada a um par de cardinalidades (mínima e máxima). No exemplo da *Figura 3.1*, a ligação $AUTOR$-R_1 tem cardinalidade

mínima igual a 1 e cardinalidade máxima igual a N. Isso quer dizer que uma instância de ***AUTOR*** deve participar no mínimo de um relacionamento em R_1 e no máximo de N relacionamentos em R_1. A ligação ***LIVRO***-R_2 tem cardinalidade mínima igual a 1 e cardinalidade máxima igual 1, logo uma instância de ***LIVRO*** pode participar de um único relacionamento em R_2 .

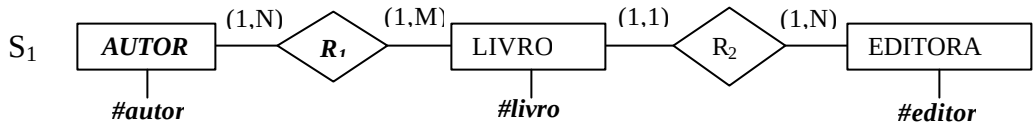


Figura 3.1 – Exemplo de ligações de relacionamento e suas respectivas cardinalidades

Definição 3.2 (*Caminho*): Se $X_1, X_2,\ldots,X_n$ são elementos de um esquema tais que $\forall i \in \{1,2,\ldots,n-1\}$, X_i-X_{i+1} é uma ligação, então X_1-X_2-…-X_n é um "***Caminho de*** X_1". □

Definição 3.3 (*Estado de Esquema*): Um estado $\mathcal{S}$ do esquema $S = (\mathcal{E}, \mathcal{R}, \mathcal{I})$, em um dado instante, é uma função definida como se segue:

- Para qualquer tipo de entidade E em $\mathcal{E}$, a extensão de E no estado $\mathcal{S}$, $\mathcal{S}(E)$, é o conjunto de entidades que são instâncias de E no estado $\mathcal{S}$.

- Para qualquer tipo de relacionamento R de $\mathcal{R}$, ligando os tipos de entidade $E_1,\ldots,E_n$, a extensão de R no estado $\mathcal{S}$, $\mathcal{S}(R)$, é o conjunto de tuplas $<e_1,\ldots,e_n>$, onde $e_i \in \mathcal{S}(E_i)$, $1 \leq i \leq n$, que são instâncias de R no estado $\mathcal{S}$. Se $r = <e_1,\ldots,e_n> \in \mathcal{S}(R)$, então dizemos que "$e_i$ *está ligada com* r através da ligação E_i-R" e "r *está ligado com* e_i através da ligação R - E_i", $1 \leq i \leq n$.

- Para qualquer atributo A de um tipo T de S (um tipo pode ser um tipo de entidade ou um tipo de relacionamento), o valor do atributo A para uma instância t de T no estado $\mathcal{S}$, $t.\mathcal{S}(A)$, é o conjunto de valores em $Dom(A)$ atribuídos a t no estado $\mathcal{S}$. Se A é um atributo monovalorado, $t.\mathcal{S}(A)$ é um conjunto unitário. Se $v \in t.\mathcal{S}(A)$, então dizemos que "t *está ligada com* v através da ligação de atributo T-A".

❖ Para qualquer caminho $C = X_1\text{-}X_2\text{-}\ldots\text{-}X_n$, onde X_i é um tipo de S, $1 \le i \le n\text{-}1$, e X_n é um tipo de S ou um atributo de X_{n-1}, o valor do caminho C para uma instância x_1 de X_1 no estado $\mathcal{I}$ é dado por:

$x_1.\mathcal{I}(C) = \{ x_n \mid x_n \in \mathcal{I}(X_n) \wedge$ (existem $x_2 \in \mathcal{I}(X_2), \ldots, x_{n-1} \in \mathcal{I}(X_{n-1})$ tais que

$\forall i \in \{1,2,\ldots,n\text{-}1\}$, x_i *está ligada com* x_{i+1} através da ligação $X_i\text{-}X_{i+1}\}$.

Se $x_n \in x_1.\mathcal{I}(C)$ então dizemos que "x_1 *está ligada com* x_n *através do caminho* C ".

C é um caminho monovalorado se uma instância de X_1 pode estar ligada a no máximo uma instância de X_n através de C.

❖ Para qualquer tipo T de S e predicado P, $\mathcal{I}(T[P]) = \{t \mid t \in \mathcal{I}(T) \wedge P(t) = true \}$.

❖ Para qualquer tipo T de S, onde $C_1,\ldots,C_n$ são caminhos de T,

$\mathcal{I}(T[C_1,\ldots,C_n]) = \{<t_1,\ldots,t_n> \mid$ existe $t \in \mathcal{I}(E)$ tal que $\forall i \in \{1,2,\ldots,n\}, t_i \in t.\mathcal{I}(C_i)\}$. □

No restante deste trabalho, quando possível, omitiremos as referências ao estado corrente $\mathcal{I}$. Por exemplo, usaremos $t.C_i$ ao invés de $t.\mathcal{I}(C_i)$. A seguir definimos os conceitos de *Tipos de Entidade Funcionalmente Equivalentes* e *Identificador de Tipo*.

Definição 3.4 (*Tipos de Entidade Funcionalmente Equivalentes*): Dois tipos de entidade E_1 e E_n são funcionalmente equivalentes ($E_1 \leftrightarrow E_n$) com respeito a um caminho $C = E_1\text{-}R_1\text{-}\ldots\text{-}E_n$, se:

(i) uma instância de E_1 pode estar ligada com no máximo uma instância de E_n através do caminho C; e

(ii) uma instância de E_n pode estar ligada com no máximo uma instância de E_1 através do caminho C. □

Definição 3.5 (*Identificador de Tipo*): Um conjunto mínimo de caminhos de um tipo T, que identificam uma única instância de T, é chamado um identificador de T ($Id(T)$). □ Se $\{C_1,\ldots,C_n\}$ é um identificador do tipo T, então para quaisquer instâncias t_1 e t_2 em T, se

$t_1.C_i = t_2.C_i$, $1 \leq i \leq n$, então $t_1 \equiv t_2$; ou seja t_1 e t_2 são semanticamente equivalentes (representam o mesmo objeto do mundo real). Note que esta definição mais genérica de conceito de identificador permite que o identificador de um tipo ***T*** possa conter atributos próprios de ***T*** (uma ligação de atributo é um caminho), assim como atributos de outros tipos que estão relacionados com ***T*** através de um caminho. Este é o caso do identificador de um tipo de relacionamento.

3.2 Assertivas de Correspondência

Para estabelecer as correspondências entre os componentes de esquemas utilizamos Assertivas de Correspondência, que são tipos especiais de restrições de integridade usadas para especificar que a semântica de algumas partes de um esquema está de alguma forma relacionada com a semântica de algumas partes de outro(s) esquema(s). Podemos classificar as assertivas em quatro grupos, definidos a seguir.

- *Assertivas de Correspondência de Tipos*
- *Assertivas de Correspondência de Atributos*
- *Assertivas de Correspondência de Caminhos*
- *Assertivas de Dependência Existencial*

No resto deste capítulo, considere o esquema $S = (E, A, J)$, onde T, $T_1,\ldots,T_n$ são tipos de S e J é um estado de S. Nas próximas seções, descrevemos cada um desses tipos de assertivas de correspondência.

3.2.1 Assertivas de Correspondência de Tipos

As assertivas de correspondência de tipos representam os relacionamentos existentes entre as extensões dos tipos de um esquema. A tabela da *Figura 3.2* contém a definição das assertivas básicas de correspondência de tipos.

As assertivas de correspondência de tipos particionam os tipos de um esquema em classes de equivalência. Os tipos T_1 e T_2 estão na mesma classe de equivalência se existe uma função de mapeamento ($\mathcal{F}$) um-para-um entre as instâncias de T_1 e as instâncias de T_2. Em geral, T_1 e T_2 têm um identificador comum, o qual é usado como função de mapeamento. Se uma instância t_1 de T_1 é mapeada na instância t_2 de T_2 ($\mathcal{F}(t_1) = t_2$), então t_1 e t_2, são "semanticamente equivalentes" ($t_1 \equiv t_2$).

RELAÇÃO	NOTAÇÃO	CONDIÇÃO PARA VALIDAÇÃO EM $\mathcal{A}$
Subconjunto	$T_1 \subset T_2$	Existe uma função injetiva $\mathcal{I}: \mathcal{A}(T_1) \rightarrow \mathcal{A}(T_2)$
Equivalência	$T_1 \equiv T_2$	Existe uma função bijetiva $\mathcal{I}: \mathcal{A}(T_1) \rightarrow \mathcal{A}(T_2)$
Disjunção	$T_1 \mid T_2$	$\neg\exists\ (t_1 \in \mathcal{A}(T_1) \ \wedge\ t_2 \in \mathcal{A}(T_2) \ \wedge\ t_1 \equiv t_2)$
Diferença	$T \equiv T_1 - T_2$	Existe uma função bijetiva $\mathcal{I}: \mathcal{A}(T) \rightarrow \mathcal{A}(T_1) - \mathcal{A}(T_2)$
União	$T \equiv \bigcup_{i=1}^{n} T_i$	Existe uma função bijetiva $\mathcal{I}: \mathcal{A}(T) \rightarrow \bigcup_{i=1}^{n} \mathcal{A}(T_i)$
Interseção	$T \equiv \bigcap_{i=1}^{n} T_i$	Existe uma função bijetiva $\mathcal{I}: \mathcal{A}(T) \rightarrow \bigcap_{i=1}^{n} \mathcal{A}(T_i)$
Seleção	$T_1 \equiv T_2[\mathcal{P}]$	Existe uma função bijetiva $\mathcal{I}: \mathcal{A}(T_1) \rightarrow \mathcal{A}(T_2[\mathcal{P}])$
Sobreposição	$T_1 \cap T_2$	Existe uma função injetiva parcial $\mathcal{I}: \mathcal{A}(T_1) \rightarrow \mathcal{A}(T_2)$

Figura 3.2 - Assertivas de Correspondência de Tipos

Outros tipos de assertivas de correspondência de tipos podem ser definidas a partir das assertivas básicas. Dizemos que T é a generalização de $T_1,\ldots,T_n$, ($T \equiv$ Gen($T_1,\ldots,T_n$)), se $T \equiv \bigcup_{i=1}^{n} T_i$ e $T_i \mid T_j$, $1 \leq i \neq j \leq n$.

3.2.2 Assertivas de Correspondência de Atributos

As assertivas de correspondência de atributos representam a equivalência semântica de atributos pertencentes a tipos "semanticamente equivalentes".

❖ *Atributos Semanticamente Equivalentes*

Definição 3.6 (*Atributos com Domínios Compatíveis*): Seja A_1 um atributo do tipo T_1 e A_2 um atributo do tipo T_2. A assertiva de correspondência $A_1 \equiv A_2$ é válida em um estado $\mathcal{I}$ se para quaisquer $t_1 \in \mathcal{I}(T_1)$ e $t_2 \in \mathcal{I}(T_2)$, se $t_1 \equiv t_2$ então $t_1.\mathcal{I}(A_1) = t_2.\mathcal{I}(A_2)$.□

Definição 3.7 (*Atributos com Domínios Diferentes*): Seja A_1 um atributo do tipo T_1 e A_2 um atributo do tipo T_2 e a função de mapeamento de domínio, γ: $Dom(A_2) \rightarrow Dom(A_1)$. A assertiva de correspondência $A_1 \equiv \gamma \circ A_2$ é válida em um estado $\mathcal{I}$ se para quaisquer $t_1 \in \mathcal{I}(T_1)$ e $t_2 \in \mathcal{I}(T_2)$, se $t_1 \equiv t_2$ então $t_1.\mathcal{I}(A_1) = \gamma\,(t_2.(\mathcal{I}(A_2)))$.□

❖ *Atributos de Definição de Generalização*

Definição 3.8: Suponha que T é uma generalização de $T_1,\ldots,T_n$ ($T \equiv \text{Gen}(T_1,\ldots,T_n)$). Seja A um atributo monovalorado de T. A assertiva de correspondência $A \equiv [T_1{:}v_1,\ldots,T_n{:}v_n]$, onde $Dom(A) = \{v_1,\ldots,v_n\}$, é válida em $\mathcal{I}$ se para quaisquer $t \in \mathcal{I}(T)$ e $t' \in \mathcal{I}(T_i)$, para algum i ∈ {1,…,n}, se $t' \equiv t$ então $t.\mathcal{I}(A) = v_i$.□

3.2.3 Assertivas de Correspondência de Caminhos

Definição 3.9: Considere $C_1 = X_1\text{-}\ldots\text{-}X_n$ e $C_2 = Y_1\text{-}\ldots\text{-}Y_n$ caminhos dos tipos X_1 e Y_1, onde X_1 e Y_1 são "semanticamente equivalentes", assim como X_n e Y_n. A assertiva de correspondência $C_1 \equiv C_2$ é válida em um estado $\mathcal{I}$ se para quaisquer $x \in \mathcal{I}(X_1)$ e $y \in \mathcal{I}(Y_1)$, se $x \equiv y$ então $x.\mathcal{I}(C_1) = y.\mathcal{I}(C_2)$.□

A seguir, veremos um exemplo que demonstra o uso das assertivas de correspondência de caminhos.

Exemplo 3.1:

Suponha os esquemas da *Figura 3.3*. A assertiva de correspondência de caminhos $EMP_1\text{-}R_1\text{-}GER_1\text{-}nger_1 \equiv EMP_2\text{-}R_2\text{-}DEPTO\text{-}R_3\text{-}GER_2\text{-}nger_2$ especifica que o nome do gerente de um empregado no esquema S_1 é equivalente ao nome do gerente do departamento deste empregado no esquema S_2.

É importante notar que, as assertivas de correspondência de atributos são um caso particular de assertivas de correspondência de caminhos, quando os caminhos são compostos de uma única ligação, sendo esta uma ligação de atributo.

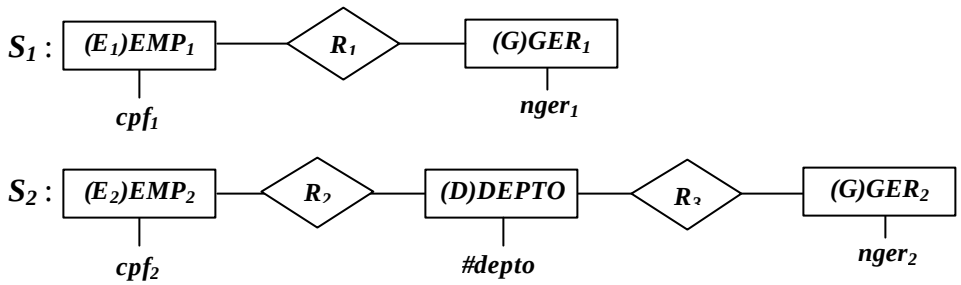


Figura 3.3- Caminhos "semanticamente equivalentes"

Uma assertiva de correspondência de caminhos da forma $T\text{-}A \equiv Y_1\text{-}Y_2\text{-}\ldots\text{-}Y_n\text{-}A'$, onde A e A' são atributos, especifica o *caminho de derivação* do *atributo derivado* A, onde Y_1 é o *tipo base* do *caminho de derivação*. Um atributo é chamado de *derivado* se ele for semanticamente equivalente a um caminho (composto de mais de uma ligação). Esse caminho é chamado de *caminho de derivação* do atributo. Um *atributo derivado* e seu *caminho de derivação* têm sempre a mesma cardinalidade (ex: se o atributo derivado é monovalorado então seu caminho de derivação também é monovalorado).

No esquema S_1 da *Figura 3.4*, $nger_1$ é um atributo derivado cujo caminho de derivação é determinado pela assertiva de correspondência de caminhos $EMP_1\text{-}nger_1 \equiv EMP_2\text{-}R_1\text{-}DEPTO\text{-}R_2\text{-}GER_2\text{-}nger_2$, que especifica que o nome do gerente de um empregado no esquema S_1 é equivalente ao nome do gerente do departamento deste empregado no esquema S_2.

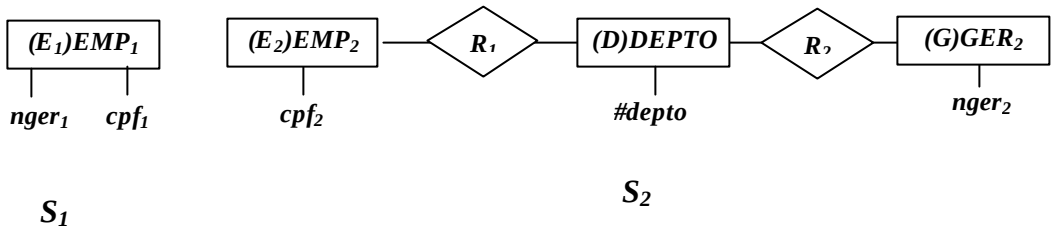


Figura 3.4 – Um atributo e um caminho são "semanticamente equivalentes"

O caminho de derivação de um atributo derivado especifica os tipos de relacionamento envolvidos nas operações de junção para determinar o valor do atributo derivado. O tipo de relacionamento ***R***, resultante da junção de todos os tipos de relacionamento que participam do caminho de derivação de um atributo ***A***, é chamado de *tipo de relacionamento de derivação* de ***A***. Uma definição formal para *tipo de relacionamento de derivação* de um atributo derivado é apresentada abaixo:

Definição 3.10 (*Tipo de Relacionamento de Derivação de Atributo Derivado*) : Seja ***A*** um atributo derivado do tipo ***T***. O tipo de relacionamento de derivação de ***A*** é definido como se segue:

Caso 1: *O tipo base do caminho de derivação de **A** é um tipo de entidade*
Suponha que a assertiva de correspondência de caminhos que especifica o caminho de derivação de ***A*** seja: $T\text{-}A \equiv E_1\text{-}R_1\text{-}\ldots\text{-}R_{n-1}\text{-}E_n\text{-}A'$, onde $E_1 \ldots E_n$ são tipos de entidade e $R_1,\ldots,R_{n-1}$ são tipos de relacionamento. Nesse caso, o tipo de relacionamento de derivação de ***A*** (R_D) é um tipo de relacionamento binário composto de E_1 e E_n dado por: $R_D = R_1 * \ldots * R_{n-1}[E_1, E_n]$. R_D é obtido fazendo-se a junção de todos os tipos de relacionamento que participam do caminho de derivação de ***A*** e fazendo a projeção nos tipos de entidade E_1 e E_n.

Caso 2: *O tipo base do caminho de derivação de **A** é um tipo de relacionamento*
Suponha que a assertiva de correspondência de caminhos que especifica o caminho de derivação de ***A*** seja: $T\text{-}A \equiv R_1\text{-}E_2\ldots\text{-}R_{n-1}\text{-}E_n\text{-}A'$, onde $E_2 \ldots E_n$ são tipos de entidade e $R_1,\ldots,R_{n-1}$ são tipos de relacionamento. Nesse caso, o tipo de relacionamento de derivação

de A (R_D) é um tipo de relacionamento binário composto de E_2 e E_n dado por: $R_D = R_2*\ldots*R_{n-1}[E_2,E_n]$. R_D é obtido fazendo-se a junção de todos os tipos de relacionamento que participam do caminho de derivação de A, exceto R_1, e fazendo a projeção nos tipos de entidade E_2 e E_n. □

3.2.4 Assertivas de Dependência Existencial

As assertivas de dependência existencial (DE) são formas gerais de restrições de dependência existencial que permitem expressar formalmente a equivalência semântica de conceitos correspondentes representados de maneiras diferentes. Diferentes representações de um mesmo conceito podem acontecer ou porque o modelo suporta diferentes representações ou porque projetistas têm diferentes percepções da realidade. A multiplicidade de possíveis representações de um conceito do mundo real é chamada de relativismo semântico. Para as definições abaixo considere $C_{11},\ldots,C_{1n}$ caminhos monovalorados do tipo T_1 e $C_{21},\ldots,C_{2n}$ caminhos monovalorados do tipo T_2.

Definição 3.11: A assertiva de dependência existencial $T_1[C_{11},\ldots,C_{1n}] \subseteq T_2[C_{21},\ldots,C_{2n}]$ é válida em $\mathcal{I}$ se para qualquer $<o_1,\ldots,o_n> \in \mathcal{I}(T_1[C_{11},\ldots,C_{1n}])$ existe $<p_1,\ldots,p_n> \in \mathcal{I}(T_2[C_{21},\ldots,C_{2n}])$ tal que $o_i \equiv p_i$, $1 \leq i \leq n$. □

Definição 3.12: A assertiva de dependência existencial $T_1[C_{11},\ldots,C_{1n}] \equiv T_2[C_{21},\ldots,C_{2n}]$ é válida em $\mathcal{I}$ se $T_1[C_{11},\ldots,C_{1n}] \subseteq T_2[C_{21},\ldots,C_{2n}]$ e $T_2[C_{21},\ldots,C_{2n}] \subseteq T_1[C_{11},\ldots,C_{1n}]$ são válidas em $\mathcal{I}$. □

Definição 3.13: Suponha a assertiva de DE $T_1[C_{11},\ldots,C_{1n}] \subseteq T_2[C_{21},\ldots,C_{2n}]$.Seja t_1 uma instância do tipo T_1 e t_2 uma instância do tipo T_2, $t_1[C_{11},\ldots,C_{1n}] = t_2[C_{21},\ldots,C_{2n}]$ sss $t_1.C_i = t_2.C_i$, $\forall i \in \{1,2,\ldots,n\}$.

Outros tipos de assertivas de dependência existencial podem ser definidas de maneira semelhante às definições de assertivas de correspondência de tipos apresentadas na tabela da *Figura 3.2*.

É importante notar que as assertivas de correspondência de tipos são casos especiais da assertiva de DE, quando algumas condições são satisfeitas. O Teorema abaixo define as condições que devem ser satisfeitas para os casos de correspondência de equivalência de tipos e o exemplo 3.2 ilustra uma situação onde isso acontece.

Teorema 3.1:

Dada a assertiva de DE $T_1[C_{11},\ldots,C_{1n}] \equiv T_2[C_{21},\ldots,C_{2n}]$. Se o $Id(T_1) \subseteq \{ C_{11},\ldots,C_{1n}\}$ e o $Id(T_2) \subseteq \{C_{21},\ldots,C_{2n}\}$, então $T_1 \equiv T_2$.

Prova:

Suponha um estado $\mathcal{I}$, onde temos a assertiva de DE $T_1[C_{11},\ldots,C_{1n}] \equiv T_2[C_{21},\ldots,C_{2n}]$, tal que o $Id(T_1) \subseteq \{ C_{11},\ldots,C_{1n}\}$ e o $Id(T_2) \subseteq \{C_{21},\ldots,C_{2n}\}$. Pela assertiva acima, temos que existe uma função injetiva total f: $\mathcal{I}(T_1[C_{11},\ldots,C_{1n}]) \rightarrow \mathcal{I}(T_2[C_{21},\ldots,C_{2n}])$. Como o $Id(T_1) \subseteq \{C_{11},\ldots,C_{1n}\}$ temos que existe uma função bijetiva total g: $\mathcal{I}(T_1) \rightarrow \mathcal{I}(T_1[C_{11},\ldots,C_{1n}])$. Da mesma maneira, temos que existe uma função bijetiva total h: $\mathcal{I}(T_2[C_{21},\ldots,C_{2n}]) \rightarrow \mathcal{I}(T_2)$, porque o $Id(T_2) \subseteq \{C_{21},\ldots,C_{2n}\}$. Logo, existe uma função bijetiva total g ° f ° h : $\mathcal{I}(T_1) \rightarrow \mathcal{I}(T_2)$. Pela definição de equivalência de subconjunto, na tabela da *Figura 3.2*, temos que: $E_1 \equiv E_2$.

Exemplo 3.2: *Um tipo de entidade é modelado como um tipo de relacionamento.*

Suponha os esquemas da *Figura 3.5*. A assertiva de dependência existencial, $POSSUI[PC\text{-}P_1\text{-}cpf_1,\ PC\text{-}C_1\text{-}lic_1] \equiv PROPRIETÁRIO[P_2\text{-}cpf_2,\ P_2\text{-}lic_2]$, captura a restrição de que a existência de um relacionamento entre uma pessoa ***p*** e um carro ***c*** em ***POSSUI*** requer a existência de um proprietário ***t*** tal que $p.cpf_1 = t.cpf_2$ e $c.lic_1 = t.lic_2$. Como $POSSUI[PC\text{-}P_1\text{-}cpf_1, PC\text{-}C_1\text{-}lic_1] \equiv POSSUI$ e $PROPRIETÁRIO[P_2\text{-}cpf_2, P_2\text{-}lic_2] \equiv PROPRIETÁRIO$, pelo Teorema 3.1 temos que $POSSUI \equiv PROPRIETÁRIO$.

Não seria possível definir formalmente a função de mapeamento entre os tipos ***POSSUI*** e ***PROPRIETÁRIO*** sem o uso da DE. Outros enfoques não definem formalmente como ocorre o mapeamento entre os tipos, apenas afirmam a existência da correspondência entre eles, sem contudo formalizá-la.

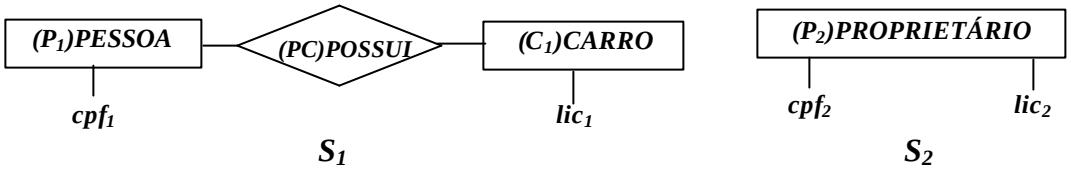


Figura 3.5- Correspondência de tipo de entidade e tipo de relacionamento.

As restrições referenciais também são casos especiais de DE, como definido abaixo:

Definição 3.14 (*Restrição Referencial Parcial*): T_1 referencia T_2 (T_1 ® T_2) sss existe uma assertiva de DE da forma $T_1[C_{11},\ldots,C_{1n}] \subseteq T_2[C_{21},\ldots,C_{2n}]$, onde $Id(T_2) \subseteq \{ C_{21},\ldots,C_{2n}\}$. Nesse caso, temos que $T_1[C_{11},\ldots,C_{1n}] \subseteq T_2$.

Definição 3.15 (*Restrição Referencial Total*): T_1 referencia T_2 (T_1 T_2) com participação total de T_2 sss existe uma assertiva de DE da forma $T_1[C_{11},\ldots,C_{1n}] \equiv T_2[C_{21},\ldots,C_{2n}]$, onde $Id(T_2) \subseteq \{ C_{21},\ldots,C_{2n}\}$. Nesse caso, temos que $T_1[C_{11},\ldots,C_{1n}] \equiv T_2$.

A seguir, mostramos alguns exemplos do uso das assertivas de DE para representar restrições referenciais.

Exemplo 3.3**:** *Um tipo de entidade é modelado com um atributo*

Suponha os esquemas da *Figura 3.6*. A assertiva de dependência existencial ***LIVRO***[***LIVRO-nome_autor***] ⊆ ***AUTOR***[***AUTOR-nome***], captura a restrição de que a existência de um livro ***l*** tal que ***l.nome_autor*** = '*Jorge Amado*' requer a existência de um autor ***a*** tal que ***a.nome*** = '*Jorge Amado*'. Nesse caso, temos que ***LIVRO*** ® ***AUTOR*** (***LIVRO***[***LIVRO-nome_autor***] ⊆ ***AUTOR***).

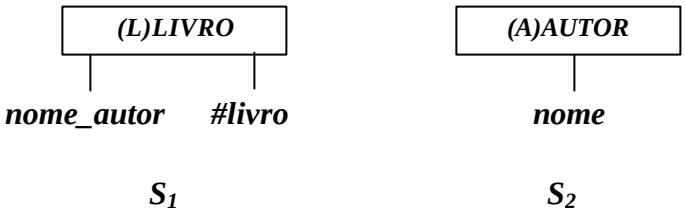


Figura 3.6 – Correspondência entre tipo de entidade e atributo

Exemplo 3.4: *Um tipo de relacionamento referencia outro tipo de relacionamento.*

Suponha os esquemas da *Figura 3.7*, a assertiva de dependência existencial, $MAT[MAT\text{-}P_1, MAT\text{-}C_1] \subseteq OFERTA[OFERTA\text{-}P_2, OFERTA\text{-}C_2]$, captura a restrição de que a existência de um relacionamento entre um professor ***p*** e um curso ***c*** em ***MAT*** requer a existência de um relacionamento entre um professor ***p*** e um curso ***c*** em ***OFERTA***. Nesse caso, temos que ***MAT*** ® ***OFERTA*** ($MAT[MAT\text{-}P_1, MAT\text{-}C_1] \subseteq OFERTA$).

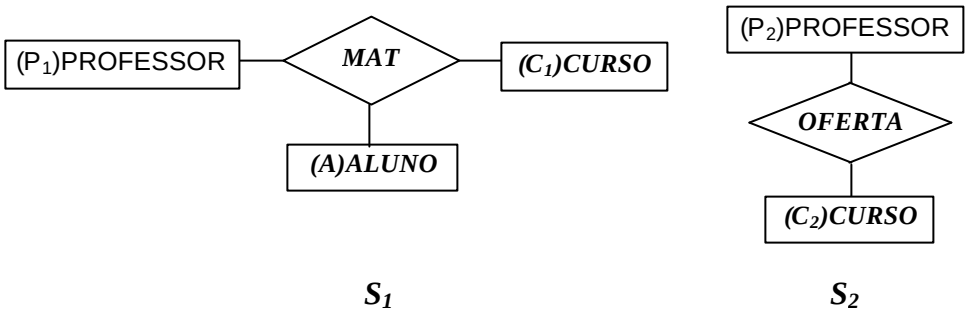


Figura 3.7 – Restrição referencial entre dois tipos de relacionamento

Exemplo 3.5: *Correspondências entre tipos de relacionamentos de graus diferentes*.

Suponha os esquemas da *Figura 3.8.* A assertiva de dependência existencial $R[R\text{-}P_1, R\text{-}E_1, R\text{-}M_1] \equiv R_1[R_1\text{-}P_2, R_1\text{-}E_2, R_1\text{-}E_2\text{-}R_2\text{-}M_2]$, captura a restrição de que a existência de um relacionamento entre um médico ***m***, uma enfermeira ***e*** e um paciente ***p*** em S_1 requer

a existência de um relacionamento entre ***p*** e ***e*** em R_1 e de um relacionamento entre ***e*** e ***m*** em R_2.

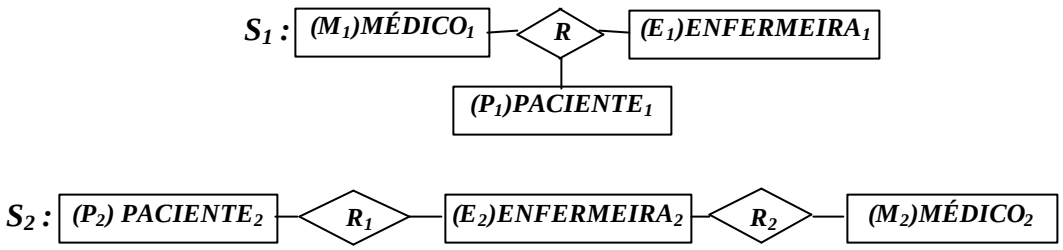


Figura 3.8 - Correspondência entre tipos de relacionamentos de graus diferentes.

Neste trabalho adotamos o modelo de entidades e relacionamentos estendido para representar o esquema do mediador e os esquemas locais. Porém, o nosso enfoque pode ser facilmente adaptado para qualquer outro modelo de dados. Uma importante contribuição deste trabalho é a extensão do modelo de Entidades e Relacionamentos com o formalismo necessário para expressar as várias formas de correspondências entre esquemas. Uma das limitações das metodologias de integração de esquemas existentes [Navathe et al.86, Spaccapietra94] é que não se pode representar a correspondência de conceitos semelhantes representados de formas diferentes. Neste trabalho, propomos formas gerais de assertivas de dependência existencial que permitem especificar formalmente os tipos mais comuns de correspondência de conceitos com múltiplas representações.

CAPÍTULO 4

METODOLOGIA PARA PROJETO DE MEDIADORES

Neste capítulo descrevemos a metodologia para projeto de mediadores que nós propomos em [Vidal97]. Esta metodologia divide o projeto de um mediador em três passos: (i) *Definição do Esquema do Mediador*, (ii) *Integração do Esquema do Mediador* e (ii) *Especificação do Mediador*. Na seção 4.1 definimos de maneira geral cada um dos passos da metodologia. Na seção 4.2 descrevemos mais detalhadamente o passo de Integração do Esquema do Mediador. Na seção 4.3 descrevemos o passo de Especificação do Mediador, o qual é dividido em dois subpassos: *Especificação do Mapeador de Estados* e *Especificação do Mapeador de Atualizações.*

4.1 Características Gerais da Metodologia

O objetivo do Projeto do Mediador é obter a especificação do mediador, a qual é constituída pelo mapeador de estados (σ) e pelo tradutor de atualizações (τ). Na metodologia proposta, o projeto de um mediador é dividido em três passos, como mostrado na *Figura 4.1*.

- *Passo 1* - *Definição do Esquema do Mediador*: Durante esse passo, os requisitos dos usuários do mediador são analisados e o esquema do mediador é especificado usando um modelo de dados de alto nível.

- *Passo 2* - *Integração do Esquema do Mediador*: O objetivo desse passo é fazer a integração do esquema do mediador com os esquemas locais para identificar as assertivas de correspondência que especificam formalmente o relacionamento do esquema do mediador com os esquemas locais.

- *Passo 3* - *Especificação do Mediador*: A partir do esquema do mediador e das assertivas de correspondência, define-se o mapeador de estados e o mapeador de atualizações do mediador. O mapeador de estados especifica como o estado dos bancos de dados locais em um determinado instante é mapeado no estado do mediador correspondente. O mapeador de atualizações especifica como atualizações requisitadas através do mediador são traduzidas em atualizações especificadas nos bancos de dados locais.

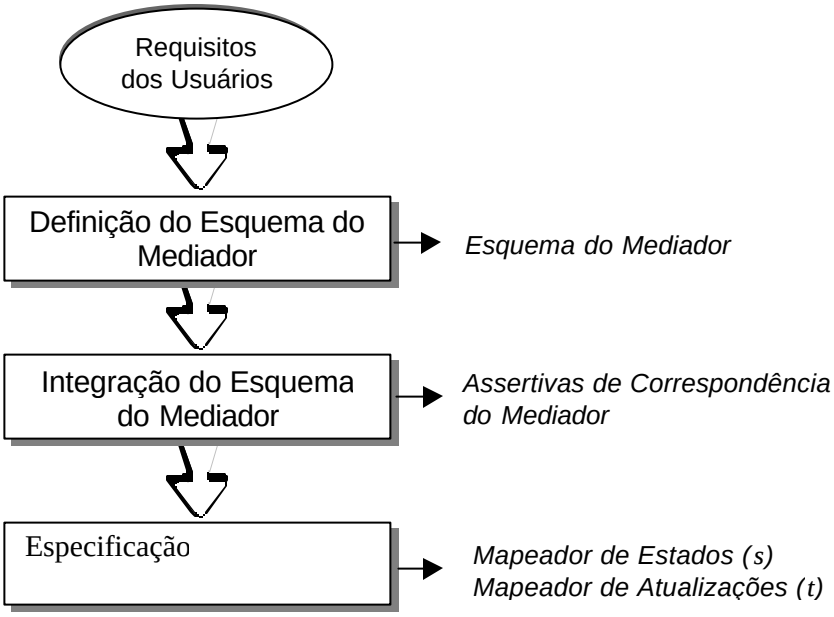


Figura 4.1 – Projeto de Mediadores

A metodologia proposta também pode ser utilizada para a definição dos mapeadores de instâncias e dos tradutores de atualizações de visões em um sistema de banco de dados centralizado. Visões constituem as interfaces através das quais os usuários consultam e atualizam o banco de dados. Uma atualização de visão é simplesmente uma atualização que é especificada em uma visão, mas que deve ser traduzida em uma seqüência de atualizações no banco de dados. O problema de *Tradução de Atualização de Visão* (TAV) refere-se à questão de definir traduções corretas de atualizações de visões. Em [Vidal96] é proposto um enfoque onde os tradutores de atualização das visões são gerados durante o passo de integração das visões do projeto do banco de dados. O enfoque proposto pela metodologia deve ser utilizado para a especificação de visões em uma diferente situação, onde o esquema do banco de dados já foi projetado e se quer adicionar uma nova visão.

4.2 Definição do Esquema do Mediador

Assim como a fase de definição de esquema no projeto tradicional de banco de dados, a fase de definição do esquema do mediador consiste em analisar os requisitos dos usuários do mediador e, em seguida, especificar o esquema do mediador usando um modelo de dados de alto nível. Neste trabalho nós adotamos o modelo de Entidades e Relacionamentos Estendido, apresentado no Capítulo 3, para a representar o esquema do mediador e os esquemas locais.

4.3 Integração do Esquema do Mediador

O objetivo da fase de Integração do Esquema do Mediador é identificar e especificar os relacionamentos entre o esquema do mediador e os esquemas locais. Existe uma grande dificuldade para usuários e projetistas encontrarem e conhecerem as informações disponíveis nos bancos de dados locais. Como mostrado na *Figura 4.2*, a metodologia propõe que seja feita, previamente, uma integração dos esquemas locais, de forma a obtermos o Esquema Global o qual contém todos os esquemas locais e um conjunto de assertivas de correspondência que especificam como os dados nos bancos de dados locais

estão relacionados (Assertivas de Correspondência Globais). Antes de efetuar a integração dos esquemas locais, é necessário mapeá-los nos esquemas ER correspondentes, caso estes não existam. É importante salientar que, no nosso enfoque, não é necessária a restruturação dos esquemas locais, como é feita em outras metodologias de integração de esquemas [Navathe96, Spaccapietra94], para a resolução de conflitos. O uso de assertivas de correspondências capazes de capturar relacionamentos entre conceitos semelhantes representados de maneiras diferentes permite o relativismo semântico.

O processo de integração do esquema do mediador com os esquemas locais consiste em especificar as assertivas de correspondência que relacionam estes esquemas (Assertivas de Correspondência do Mediador). Tais assertivas são encontradas a partir de comparações efetuadas entre objetos do esquema do mediador e objetos dos esquemas locais, para identificação das partes comuns. O processo de identificação de conceitos similares não é uma tarefa fácil principalmente para grandes aplicações. Para ajudar no processo de descoberta das assertivas de correspondência, várias ferramentas já foram desenvolvidas[Navathe et al.86, Savasere et al.91].

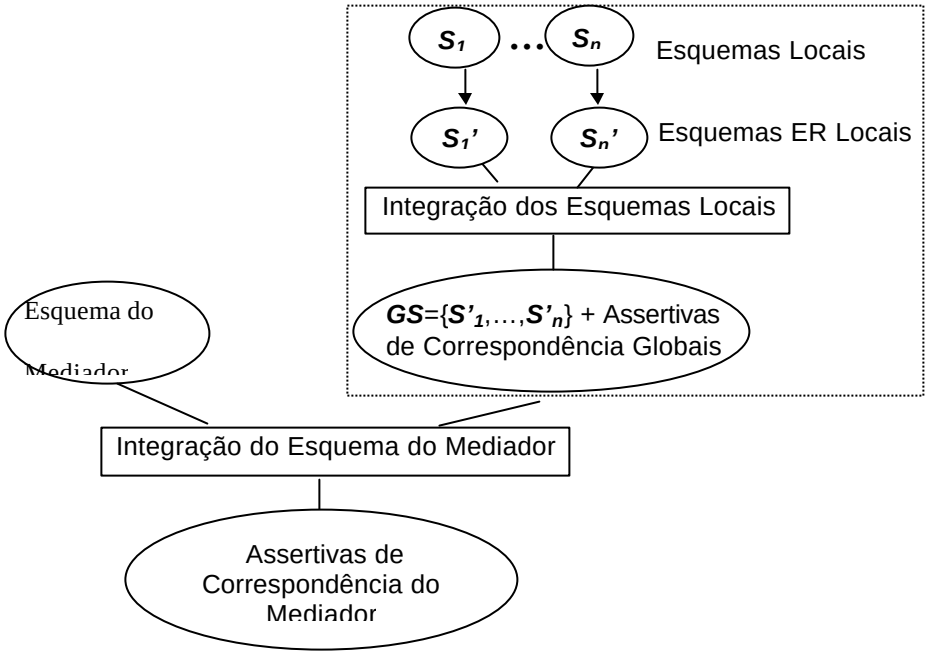


Figura 4.2- Integração de Esquemas

A identificação prévia das assertivas de correspondência globais facilitará a integração incremental do esquema do mediador com os esquemas dos bancos locais. Ao integrarmos o esquema do mediador com um dos esquemas locais, podemos inferir, com base nas assertivas de correspondência globais, novas correspondências entre o esquema do mediador e outros esquemas locais. Outra importância das assertivas de correspondência globais é que elas capturam a existência de redundância, significando que existem restrições inter-esquemas que devem ser mantidas pelas atualizações dos bancos de dados locais, uma vez que dados replicados devem ser mantidos consistentes.

4.4 Especificação do Mediador

A partir do esquema do mediador e das assertivas de correspondência que especificam formalmente os relacionamentos do esquema do mediador com os esquemas locais, define-se o mapeador de estados e o mapeador de atualizações do mediador.

Suponha um mediador $\boldsymbol{M}$ que integra informações das bases de dados $\boldsymbol{S_1,\ldots,S_n}$. O mapeador de estados do mediador $\boldsymbol{M}$ (σ_M) especifica como os estados dos esquemas locais $\mathcal{S}_{S1},\ldots,\mathcal{S}_{Sn}$, em determinado instante, são mapeados no estado do mediador correspondente, como indicado no diagrama da figura abaixo.

$$\{\mathcal{S}_{S1},\ldots,\mathcal{S}_{Sn}\} \xrightarrow{\sigma_M} \mathcal{S}_M = \sigma_M(\{\mathcal{S}_{S1},\ldots,\mathcal{S}_{Sn}\})$$

Figura 4.3 - Mapeador de Estados do mediador M.

O mapeador de estados de um mediador é definido diretamente das assertivas de correspondência que relacionam o esquema do mediador com os esquemas locais. Para a definição do mapeador de estados de $\boldsymbol{M}$, deve-se especificar para cada tipo $\boldsymbol{T_M}$ no esquema do mediador, o mapeador de estados de $\boldsymbol{T_M}$, o qual determina qual é a extensão de $\boldsymbol{T_M}$ ($\mathcal{S}_M(\boldsymbol{T_M})$) para os estados dos esquemas locais $\mathcal{S}_{S1},\ldots,\mathcal{S}_{Sn}$ ($\mathcal{S}_M(\boldsymbol{T_M}) = \sigma_M(\{\mathcal{S}_{S1},\ldots,\mathcal{S}_{Sn}\})(\boldsymbol{T_M})$). As instâncias de um tipo $\boldsymbol{T_M}$ de um mediador são obtidas da fusão de instâncias de um ou mais tipos dos esquemas locais, que são chamados *tipos base* de $\boldsymbol{T_M}$. O mapeador de

estados do tipo T_M é obtido das assertivas de correspondência que relacionam T_M com os seus *tipos base*. O valor do atributo A_M para uma instância de T_M é derivado das assertivas de correspondência de atributo e/ou de caminho que relacionam A_M com atributos e/ou caminhos de seus *tipos base.* Um atributo de mediador é chamado de "herdado" se ele for semanticamente equivalente a um atributo de um de seus tipos base, e é chamado de "derivado" se ele for semanticamente equivalente a um caminho (composto de mais de uma ligação) de um de seus *tipos base*. Se um atributo de mediador tem associado a ele algumas funções ou expressões aritméticas, então dizemos que ele é um atributo "computado". Atributos computados não podem ser atualizados através de mediadores.

O sistema de mediação MedMaker [Papakonstantinou et al.96] provê uma linguagem de alto nível chamada MSL que permite uma especificação declarativa do mapeador de estados do mediador. Nesse sistema, o mapeador de estados consiste de um conjunto de regras que definem como os objetos dos tipos do mediador são obtidos da fusão dos objetos dos seus tipos base. Os algoritmos para gerar o mapeador de estados do mediador, a partir das assertivas de correspondência, estão sendo desenvolvidos em outra dissertação de Mestrado.

O mapeador de atualizações de um mediador, também chamado de tradutor de atualizações, especifica de que maneira as atualizações definidas sobre o esquema do mediador são traduzidas em atualizações correspondentes definidas nos esquemas locais. O mapeador de atualizações de um mediador também pode ser definido diretamente das assertivas de correspondência que relacionam o esquema do mediador com os esquemas locais.

Atualmente, o uso de mediadores é restrito ao acesso integrado de informações em múltiplas bases de dados. Mas, certamente é de grande interesse que eles também sejam usados para atualização de múltiplas bases de dados. É importante salientar que, esse uso será restrito a uma classe limitada de atualizações, uma vez que existem certas atualizações de mediador, para as quais não existe uma tradução. Neste trabalho, mostramos que a partir das assertivas de correspondência do mediador podemos definir precisamente para que

tipos de atualizações existe uma tradução e, caso esta exista, qual a tradução correspondente. No próximo capítulo, discutimos o problema de definir o mapeador de atualizações de um mediador e apresentamos o nosso enfoque para definir tradutores para as operações de atualização de um mediador.

CAPÍTULO 5

ATUALIZAÇÃO DE MÚLTIPLAS BASES DE DADOS ATRAVÉS DE MEDIADORES

Neste capítulo, discutiremos o nosso enfoque para atualização de múltiplas bases de dados através de mediadores. Atualizar múltiplas bases de dados através de mediadores é muito semelhante à atualização de bancos de dados através de visões. Sendo assim, na seção 5.1 abordamos a atualização através de visões e descrevemos o processo de tradução de atualização de visão. Em seguida, na seção 5.2 discutimos o problema de atualização de múltiplas bases de dados através de mediadores e descrevemos o processo de tradução de atualização através de mediadores. Na seção 5.3 apresentamos o nosso enfoque para definir tradutores para as operações de atualização de mediadores e na seção 5.4 discutimos alguns trabalhos relacionados.

5.1 Atualização de Bancos de Dados através de Visões

Na arquitetura de três níveis de esquemas, as visões constituem esquemas externos que permitem ao usuário ignorar dados que não são relevantes a sua aplicação. Como uma visão é apenas uma interface através da qual os usuários acessam e atualizam o banco de

dados, qualquer atualização especificada na visão deve ser traduzida em atualizações a serem executadas no banco de dados. Uma definição de visão consiste de:

(i) *o mapeador de instâncias*: especifica como o estado do banco de dados em um determinado instante é mapeado no estado da visão correspondente;

(ii) *o tradutor de atualizações*: especifica como atualizações especificadas na visão são traduzidas em atualizações especificadas no banco de dados.

Uma atualização de visão é simplesmente uma atualização que é especificada em uma visão, mas que deve ser traduzida em uma seqüência de atualizações no banco de dados. O problema de *Tradução de Atualização de Visão* (TAV) refere-se à questão de definir traduções corretas de atualizações de visões. Em [Vidal96] é proposto um enfoque onde os tradutores de atualização das visões são gerados durante o passo de integração das visões do projeto do banco de dados.

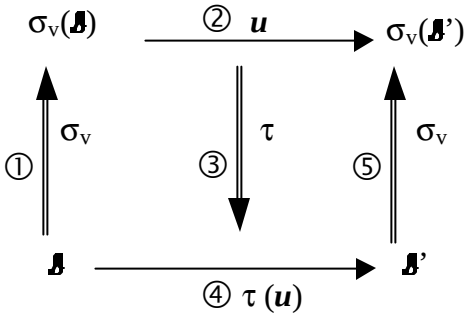


Figura 5.1 - Tradução de atualização através de visão

O processo de tradução de atualização de visão (TAV) é descrito no diagrama da *Figura 5.1*. O estado inicial do banco de dados B é mapeado pelo mapeador de instâncias σ_v no estado da visão $\sigma_v(B)$. O usuário especifica a atualização $\boldsymbol{u}$ sobre o estado da visão. A atualização de visão $\boldsymbol{u}$ deve ser traduzida em uma seqüência de atualizações sobre o banco de dados $\tau(\boldsymbol{u})$. $\tau(\boldsymbol{u})$ é executada no estado do banco de dados B para obter o novo estado do banco de dados $B'=\tau(\boldsymbol{u})(B)$. Com o novo estado do banco de dados B', obtemos o novo estado da visão correspondente $\sigma_v(B')$.

5.2 Atualização de Múltiplas Bases de Dados através de Mediadores

Atualizar dados distribuídos em múltiplas bases de dados através de um mediador é semelhante a atualizar dados através de visões em um banco de dados centralizado. Assim como as visões, os mediadores também são interfaces através das quais os usuários acessam e atualizam múltiplas bases de dados. Dessa forma, atualizações submetidas a um mediador também devem ser traduzidas em atualizações a serem executadas nos bancos de dados (BDs) correspondentes.

O processo de tradução de atualização de mediadores (TAM) é descrito pelo diagrama da *Figura 5.2.* Os estados iniciais dos BDs locais $\mathcal{S}_{S1},\ldots,\mathcal{S}_{Sn}$ são mapeados pelo mapeador de estados σ_M no estado do mediador $\mathcal{S}_M$. O usuário especifica a atualização $\boldsymbol{u_M}$ sobre o estado do mediador. A atualização $\boldsymbol{u_M}$ requisitada ao mediador deve ser traduzida em um conjunto de atualizações sobre os BDs locais $\tau_M(\boldsymbol{u_M}) = \{ \tau_{S1}(\boldsymbol{u_M}),\ldots, \tau_{Sn}(\boldsymbol{u_M})\}$, onde $\tau_{Si}(\boldsymbol{u_M})$ contém todas as atualizações em $\tau_M(\boldsymbol{u_M})$ referentes ao esquema S_i. Cada $\tau_{Si}(\boldsymbol{u_M})$ é executado no estado do banco de dados $\mathcal{S}_{Si}$ obtendo-se o novo estado $\mathcal{S}'_{Si} = \tau_{Si}(\boldsymbol{u_M})(\mathcal{S}_{Si})$. Com os novos estados dos BDs locais $\mathcal{S}'_{S1},\ldots, \mathcal{S}'_{Sn}$, obtém-se o novo estado do mediador correspondente $\mathcal{S}'_M = \sigma_M(\{\mathcal{S}'_{S1},\ldots, \mathcal{S}'_{Sn}\})$.

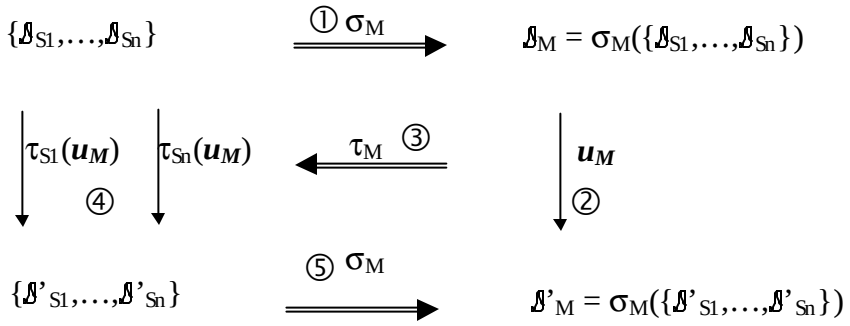


Figura 5.2 - Tradução de atualização através de mediador

Algumas vezes uma tradução de uma atualização de mediador (u_M) pode causar mudanças adicionais no mediador além dos efeitos diretos desejado por u_M. Estas mudanças adicionais são chamadas de *"efeitos colaterais"*. Além disso, como sabemos, uma atualização em um banco de dados pode disparar outras atualizações, as quais são necessárias para corrigir violações de restrições de integridade do esquema do banco de dados. Assim sendo, o novo estado do mediador vai depender também dos estados dos BDs locais e de suas restrições de integridade.

Um pedido de atualização submetido ao mediador pode ter nenhuma, uma, ou múltiplas traduções. Uma das dificuldades associada ao problema de TAM surge quando existe mais de uma tradução possível, e apenas uma deve ser escolhida. Existem algumas formas de ambigüidades que podem ser resolvidas em *"tempo de definição"* do medidor. É o que chamamos de ambigüidades a *"nível de esquema"*, uma vez que estas são causadas pela existência de mais de um componente do esquema onde se pode realizar a atualização. Nesse caso, a escolha do componente mais apropriado pode ser feita em tempo de definição do mediador. No nosso enfoque, a preferência é dada para a tradução que cause o mínimo de efeitos colaterais no mediador.

No caso das ambigüidades a *"nível de dados"*, que são causadas pela existência de mais de uma instância onde se pode realizar a atualização, não podemos resolvê-las em tempo de projeto. Estas formas de ambigüidades só podem ser resolvidas em tempo de

atualização, e para resolvê-las se faz necessário um diálogo com o usuário (em tempo de atualização) para que ele escolha a instância mais apropriada.

No nosso enfoque, o tradutor para cada tipo de operação de atualização do mediador é definido em tempo de projeto. Uma vez definido o tradutor, o usuário especifica atualizações através do mediador e o tradutor as traduz em atualizações nos BDs locais, sem a necessidade de qualquer diálogo adicional. Na próxima seção, mostramos como os tradutores são definidos, baseado apenas nas assertivas de correspondência do mediador e na semântica de atualização dessas assertivas, as quais são fornecidas pelo projetista do mediador.

5.3 Definindo Tradutores para as Operações de Atualização de Mediadores

No nosso enfoque, o *mapeador de atualizações* de um mediador consiste de um conjunto de *tradutores*, um para cada uma das operações de atualização de mediador que forem permitidas. Um tradutor é uma função que recebe como entrada um pedido de atualização $\boldsymbol{u_M}$ e gera a *tradução* para $\boldsymbol{u_M}$, a qual consiste de uma seqüência de atualizações que devem ser realizadas nos BDs locais, de maneira que estes fiquem *consistentes* com o novo estado do mediador, supondo-se que $\boldsymbol{u_M}$ fosse realizado diretamente no mediador. As atualizações requeridas nos BDs locais são estas necessárias para preservação das assertivas de correspondência do mediador. As assertivas de correspondência do mediador são tratadas como restrições de Integridade que devem ser preservadas pelas operações de atualização do mediador.

O problema de definição do tradutor para uma dada operação de atualização de mediador, consiste portanto, em determinar a seqüência de atualizações nos BDs locais que são requeridas para a manutenção das assertivas de correspondência do mediador "relevantes" para esta operação.

Neste trabalho, desenvolvemos algoritmos, apresentados no Apêndice A, que geram tradutores para as seguintes operações de atualização de mediadores:

1. Atualização de Atributos

- **Algoritmo A1**: Gera o tradutor para a modificação de um atributo de mediador

2. Atualização de Tipos de Entidade

- **Algoritmo E1**: Gera o tradutor para a Adição em um Tipo de Entidade de Mediador

- **Algoritmo E2**: Gera o tradutor para a Remoção em um Tipo de Entidade de Mediador

3. Atualização de Tipos de Relacionamento de Mediador

- **Algoritmo R1**: Gera o tradutor para a Adição em um Tipo de Relacionamento de Mediador

- **Algoritmo R2**: Gera o tradutor para a Remoção em um Tipo de Relacionamento de Mediador

Nos algoritmos propostos, os tradutores são gerados a partir das assertivas de correspondência (AC) do mediador. Na definição de um tradutor para uma dada operação de atualização, o primeiro passo é determinar as AC relevantes, isto é as AC que podem ser violadas por essa operação. Para cada AC relevante deve-se então determinar a ação requerida para a sua preservação, isto é como tornar a AC válida supondo-se que a atualização tenha sido realizada no mediador. As ações requeridas para a manutenção de

assertivas de correspondência de tipo e de dependência existencial, estão definidas em tabelas apresentadas no Apêndice B, as quais são :

- *Tabela 1*: define as ações requeridas para a manutenção das assertivas de correspondência de tipos para a adição em um tipo mediador.

- *Tabela 2*: define as ações requeridas para a manutenção das assertivas de correspondência de tipos para a remoção em um tipo de mediador.

- *Tabela 3*: define as ações requeridas para a manutenção das assertivas de dependências existenciais para a adição em um tipo de mediador.

- *Tabela 4*: define as ações requeridas para a manutenção das assertivas de dependências existenciais para a remoção em um tipo de mediador.

No caso das assertivas de correspondência em que existe mais de uma maneira de fazer a manutenção, escolhe-se a opção que cause menos efeitos colaterais no mediador. Nos casos em que as opções disponíveis não fazem diferença em termos de efeitos colaterais, o algoritmo consultará o projetista para que ele forneça uma informação extra(semântica de atualização - ***SA***) para indicar qual a opção a ser utilizada. Este é o caso da assertiva de correspondência de tipo $\boldsymbol{E_M}$ **Ì** ***E***, na Tabela 1 do Apêndice B, onde a ***SA*** pode ser “Bloqueia” ou “Propaga”. Este tipo de ambigüidade caracteriza-se como ambigüidade a nível de esquema, portanto pode ser resolvida em tempo de definição do mediador. Por outro lado, nos casos de ambigüidades a nível de dados, estas que só podem ser resolvidas em tempo de atualização, o tradutor não poderá ser definido.

5.4 Trabalhos Relacionados

A atualização de múltiplas bases de dados através de mediadores é um problema que ainda não foi abordado na literatura atual. As pesquisas atuais têm focalizado o uso de mediadores apenas para o acesso integrado a informações distribuídas em múltiplas bases de dados. Porém, como o problema de atualização de múltiplas bases de dados através de

mediadores é muito semelhante ao problema de atualização de bancos de dados através de visões, alguns resultados obtidos nas pesquisas desenvolvidas para tratar a atualização através de visões podem ser aplicados para a atualização através de mediadores.

O problema de atualização de bancos de dados através de visões tem sido estudado por muitos pesquisadores, incluindo [Bancilhon81, Dayal82, Masunaga84, Furtado85, Medeiros85, Keller86a, Larson91, Ling96]. Em [Furtado85] os trabalhos anteriores em atualizações de visões são classificados em duas abordagens. A primeira abordagem é baseada no tratamento de visões como tipos de dados abstratos, onde a definição da visão inclui todas as atualizações de visões permitidas juntamente com seus tradutores. A outra abordagem consiste em definir procedimentos gerais de tradução [Dayal82, Bancilhon81, Larson91]. Estes procedimentos recebem como entrada a definição da visão, uma atualização de visão e o estado atual do esquema. Eles produzem, se possível, uma tradução da atualização de visão em atualizações no esquema conceitual, que satisfaçam algumas propriedades. O enfoque de definir procedimentos gerais de tradução [Dayal82, Bancilhon81] somente pode ser aplicado a uma classe restrita de atualização de visões, porque alguns tipos de atualizações de visões requerem que mais semântica seja fornecida pelo usuário, para eliminar ambigüidades nas traduções das atualizações em visões [Keller86a, Medeiros85].

Outro ponto de divergência entre os enfoques propostos é no que se refere ao tratamento das ambigüidades. Os enfoques mais antigos não tratam as ambigüidades, uma vez que esses enfoques só consideram possível gerar uma tradução nos casos em que existir uma única tradução possível. As demais abordagens consideram que uma atualização pode ter mais de uma tradução possível e que as ambigüidades na tradução podem ser resolvidas usando informações semânticas adicionais. Algumas dessas abordagens tratam as ambigüidades apenas a nível de esquema, como é o caso de [Keller86a] e [Ling96]. Outros enfoques tratam as ambigüidades tanto a nível de esquema quanto a nível de dados [Larson91].

Entre os enfoques que tratam o problema de atualização através de visões e consideram que uma atualização pode ter mais de uma tradução possível, destacamos [Ling96], [Larson91] e [Keller86a], os quais são descritos a seguir.

Em [Keller86a] é proposto um método para atualizar bancos de dados relacionais através de visões. Nesse enfoque, a definição do tradutor é armazenada juntamente com a definição da visão. O enfoque de Keller propõe que a semântica necessária para remover ambigüidades na tradução das atualizações seja obtida através de diálogo com o usuário na hora da definição da visão. Nesse enfoque, as visões são definidas pelo administrador do banco de dados (DBA), que através de uma ferramenta para definição de visões fornece a semântica necessária para escolher o tradutor mais adequado. Esse enfoque só pode ser aplicado para uma classe limitada de visões denominadas “independente dos dados”, pois são visões cujas atualizações podem ser definidas antecipadamente na definição da visão sem nenhuma consideração dos dados.

Em [Larson91] é proposta uma abordagem baseada em regras para atualização de relações através de visões. Essa abordagem procura expandir os limites de outros enfoques tratando visões definidas a partir de operadores como União, Diferença, etc. Larson trata as operações de atualização de adição e remoção, entretanto não aborda a tradução para as operações de modificação, que sem dúvida nenhuma é um problema bastante complexo. Essa abordagem usa tanto o conhecimento do esquema da visão como o conhecimento semântico para decidir a tradução apropriada para uma atualização e resolve ambigüidades tanto a nível de esquema como a nível de dados.

Em [Ling96] é abordado o problema de atualizações de visões em sistemas de gerenciamento de banco de dados baseados no modelo ER [Ling87], onde visões são modeladas por diagramas ER. Ling desenvolveu uma teoria dentro da abordagem do modelo ER que caracteriza as condições sobre as quais existe mapeamento de atualizações de visões em atualizações no esquema conceitual. Baseado nesta teoria, foram desenvolvidos algoritmos que sistematicamente determinam se tipos de entidade e tipos de relacionamento podem ser atualizados. Também foram desenvolvidos algoritmos que

traduzem um pedido de atualização de visão em uma atualização no banco de dados correspondente baseado nos resultados obtidos dos algoritmos que determinam a capacidade de atualização dos tipos de entidade e tipos de relacionamento. Ling resolve as ambigüidades a nível de esquema sem consultar o usuário, nessa abordagem as ambigüidades são resolvidas pelos próprios algoritmos que geram as traduções. Os critérios utilizados na resolução das ambigüidades não são discutidos no artigo.

A nossa abordagem para o problema de atualização de múltiplas bases de dados através de mediadores consiste em gerar tradutores para as operações básicas de atualização de mediadores, os quais são armazenados juntamente com a definição do mediador. Assim, uma vez definido o tradutor, o usuário especifica atualizações através do mediador e o tradutor as traduz em atualizações nos BDs locais, sem a necessidade de qualquer diálogo adicional.

Gerar traduções a partir de tradutores específicos ao invés de usar procedimentos gerais de tradução é muito mais simples e eficiente. Com o uso de tradutores os conhecimentos sobre o esquema do mediador e as assertivas de correspondência que relacionam o esquema do mediador com os bancos de dados locais só precisam ser analisados uma única vez, na hora em que o tradutor é gerado.

Quanto ao tratamento das ambigüidades, o nosso enfoque é semelhante ao de [Keller86a]: nós tratamos apenas as ambigüidades que podem ser resolvidas em tempo de definição. As ambigüidades que só podem ser resolvidas em tempo de atualização não são tratadas, porque não achamos conveniente dar essa atribuição ao usuário do mediador, uma vez que este, a princípio, não tem conhecimento sobre os esquemas dos bancos de dados locais. Embora tenhamos feito esta opção, isto não impede que os nossos algoritmos sejam adaptados para que ambigüidades a nível de dados também sejam resolvidas pelos tradutores em tempo de atualização.

Os algoritmos propostos na nossa abordagem geram tradutores para as operações básicas de atualização de mediadores a partir das assertivas de correspondência do mediador. A vantagem do uso de assertivas de correspondência, para especificar formalmente os relacionamentos entre o esquema do mediador e os esquemas dos BDs

locais, é que nos permite provar formalmente que os tradutores gerados pelos nossos algoritmos produzem traduções corretas. A partir das assertivas de correspondência e de suas semânticas de atualização (***SA***) é possível definir precisamente qual o efeito que uma atualização submetida ao mediador deve ter nos bancos de dados locais. Um dos problemas de outros enfoques para TAV é que eles não usam um formalismo para representar os relacionamentos entre o esquema da visão e o esquema do banco de dados. Dessa forma, não é possível garantir que as traduções geradas pelos algoritmos propostos nesses enfoques são corretas.

CAPÍTULO 6

ATUALIZAÇÃO DE ATRIBUTOS

Neste capítulo, descrevemos o algoritmo que gera tradutores para as operações de *modificação* de atributos monovalorados de mediador. Uma operação de modificação de um atributo modifica o valor do atributo, caso este atributo já tenha um valor definido, ou então atribui um valor para o atributo, caso contrário. Na seção 6.1 descrevemos os passos do algoritmo e na seção 6.2, apresentamos alguns exemplos de tradutores para operações de modificação de atributos, que foram gerados usando o algoritmo descrito neste capítulo.

6.1 Definindo Tradutores para as Operações de Modificação de Atributos

O Algoritmo *A1*, apresentado na seção A.1 do Apêndice A, gera tradutores para as operações de modificação de atributos. Um pedido de modificação de atributo é expresso por "$t_M.A_M$: = 'v' ", onde lê-se: atribua o valor 'v' para o atributo A_M da instância t_M. O tradutor para a modificação do atributo A_M é denominado ***Modifica_*A_M**. Quando o usuário pedir para modificar o valor do atributo A_M, o tradutor ***Modifica_*A_M** será chamado e serão

passados como parâmetros para ele a instância t_M a ser modificada e o novo valor a ser atribuído a A_M.

Como visto anteriormente, o tradutor para uma operação de atualização de mediador determina uma seqüência de atualizações nos bancos de dados locais, as quais são requeridas para a manutenção das assertivas de correspondência de mediador relevantes para esta operação. No caso da operação de modificação de atributo, as assertivas de correspondência relevantes e que portanto devem ser consideradas no algoritmo *A1* são:

- *Assertivas de Correspondência de Atributos*
- *Assertivas de Correspondência de Caminhos*

O algoritmo *A1* é dividido em dois passos: o *passo 1* define as atualizações requeridas para a manutenção das assertivas de correspondência de atributos e o *passo 2* define as atualizações requeridas para a manutenção das assertivas de correspondência de caminhos.

❖ *Manutenção das Assertivas de Correspondência de Atributos*

O passo 1 do algoritmo *A1* define as atualizações requeridas para a manutenção das assertivas de correspondência de atributos. Na nossa abordagem, não permitiremos a modificação de atributos que sejam identificadores. Como os identificadores de tipos de entidade também são utilizados como identificadores dos tipos de relacionamento nos quais os tipos de entidade participam, modificações nos identificadores de tipos de entidade poderiam causar anomalias de atualização indesejáveis [Ling96].

Para a manutenção das assertivas de correspondência de atributo, a ação requerida nos bancos de dados locais consiste em determinar a instância do tipo base que deverá ser modificada e então, modificar o valor do atributo base de acordo com o valor atribuído ao atributo de mediador.

❖ *Manutenção das Assertivas de Correspondência de Caminhos*

O passo 2 do algoritmo *A1* define as atualizações requeridas para a manutenção das assertivas de correspondência de caminhos. O problema de definir atualizações para a manutenção destas assertivas é bem mais complexo e assemelha-se ao problema de atualização de atributos. As assertivas de correspondência de caminhos especificam os caminhos de derivação de atributos derivados de tipos de mediador. O caminho de derivação de um atributo derivado determina os tipos de relacionamento locais envolvidos em operações de junção para determinar o atributo derivado. Assim, o valor de um atributo derivado pode ser obtido a partir dos valores das ligações(de atributo ou relacionamento) que compõem o caminho de derivação do atributo. Logo, para modificar o valor de um atributo derivado deve-se modificar o valor de uma das ligações no caminho de derivação deste atributo, como mostrado no exemplo a seguir.

Considere o esquema local $\boldsymbol{S_1}$ e o esquema de mediador $\boldsymbol{S_2}$ da *Figura 6.1*. O valor do atributo #***proj_m*** (número do projeto) de uma instância $\boldsymbol{e_M}$ em ***EMP_M*** é modificado quando :

1. *Modifica-se o departamento que está associado com* $\boldsymbol{e_M}$ *através de* $\boldsymbol{R_1}$.

2. *Modifica-se o projeto que está associado com o departamento de* $\boldsymbol{e_M}$ *através de* $\boldsymbol{R_2}$.

3. *Modifica-se o valor do atributo* ***#proj*** da instância de ***PROJ***, *que está associada com o departamento de* $\boldsymbol{e_M}$.

Assim, para definirmos a ação requerida para um pedido de modificação de ***#proj_m***, temos que primeiro escolher qual das opções acima é a mais adequada.

O passo 2 do algoritmo *A1*, que trata as assertivas de correspondência de caminhos é dividido em dois subpassos: o *Passo 2.1* determina o tipo de relacionamento base onde deverá ser efetuada a atualização, com relação a uma determinada assertiva de correspondência de caminhos e o *Passo 2.2* determina a atualização requerida para a

manutenção de uma dada assertiva de correspondência de caminhos. Nas próximas seções discutiremos cada um desses passos.

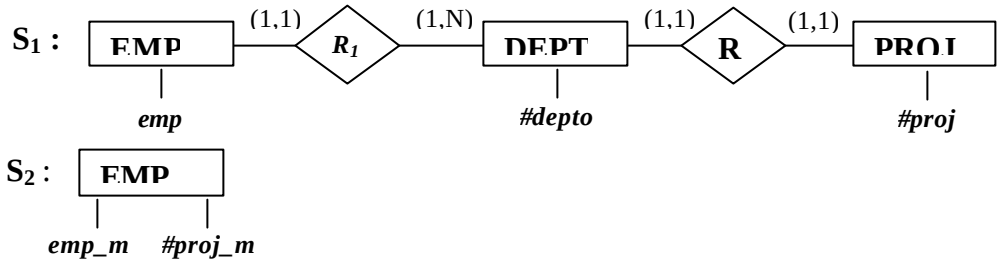


Figura 6.1 – Esquema local S_1 e Esquema de Mediador S_2

6.1.1 *Passo 2.1 :* Determina o Tipo de Relacionamento onde deverá ser efetuada a atualização

No passo 2.1, determina-se o tipo de relacionamento no caminho de derivação do atributo, no qual deverá ser efetuada a atualização. Falamos apenas em termos de tipos de relacionamento, porque como veremos a seguir, a ligação a ser modificada deve ser uma ligação de relacionamento.

Na nossa abordagem, estabelecemos os seguintes critérios para a escolha da ligação onde deve ser feita a modificação:

(i) *A modificação deve ser realizada em uma ligação de relacionamento.*

O motivo dessa restrição é que não deve ser permitido modificar o valor de um atributo de uma entidade base que não seja semanticamente equivalente à entidade de mediador que está sendo atualizada. É o caso da 3ª. opção do exemplo anterior, onde uma instância de empregado é modificada quando modificamos um atributo de uma instância de projeto, visto que empregado não é semanticamente equivalente a projeto.

Assim, temos que a ligação a ser atualizada deve ser uma ligação de relacionamento. Dessa forma, dada a assertiva de correspondência de caminhos $\boldsymbol{Y_M}$-

$A_M \equiv Y_1\text{-}Y_2\text{-}\ldots\text{-}Y_n\text{-}A$, a ação requerida para a modificação de A_M é a modificação de um relacionamento em um dos tipos de relacionamento que fazem parte do caminho de derivação de A_M. No exemplo anterior, isso acontece quando modifica-se o atributo #***proj_m*** pela modificação de um relacionamento em R_1 ou de um relacionamento em R_2. Para que seja possível realizar a atualização em uma ligação de relacionamento é necessário que A seja um identificador de Y_n, pois isto garante que com o valor de A iremos recuperar apenas uma instância de Y_n.

(ii) *O tipo de relacionamento base onde será efetuada a atualização deve ser "semanticamente equivalente" ao tipo de relacionamento de derivação do atributo.*

Este critério é desejável para que uma atualização em um banco de dados local cause o mínimo de efeitos colaterais na visão do mediador. O Teorema 6.1 abaixo estabelece as condições necessárias e suficientes para determinarmos quais os tipos de relacionamento no caminho de derivação de um atributo, que são semanticamente equivalentes ao tipo de relacionamento de derivação deste atributo.

Teorema 6.1:

Suponha o caminho de derivação do atributo A dado por: $E_1\text{-}R_1\text{-}\ldots\text{-}E_i\text{-}R_i\text{-}E_{i+1}\text{-}\ldots\text{-}R_{n-1}\text{-}E_n\text{-}A$. Seja R o tipo de relacionamento de derivação de A. R é semanticamente equivalente a R_i ($R \equiv R_i$) sss

(a) E_i é funcionalmente equivalente a E_1 ($E_i \leftrightarrow E_1$)

(b) E_{i+1} é funcionalmente equivalente a E_n ($E_n \leftrightarrow E_{i+1}$)

(c) $\{R_i\text{-}E_i, R_i\text{-}E_{i+1}\}$ é um identificador de R_i

Prova: Temos que mostrar que $R \subseteq R_i$ e $R_i \subseteq R$

(1) $R \subseteq R_i$ sss para qualquer $r \in R$ então

(i) Existe $r_i \in R_i$ tal que $r_i.(R_i\text{-}E_i\text{-}\ldots\text{-}E_1) \equiv r.(R\text{-}E_1)$ e $r_i.(R_i\text{-}E_{i+1}\text{-}\ldots\text{-}E_n) \equiv r.(R\text{-}E_n)$

(ii) r_i é único

(i) segue direto da definição de R ($R = R_1 * \ldots * R_{n-1}[E_1, E_n]$).

Suponha r onde $r.(R\text{-}E_1) = e_1$ e $r.(R\text{-}E_n) = e_n$. Como $E_1 \leftrightarrow E_i$ então só pode existir uma instância $e_i \in E_i$ tal que $e_i.(E_i\text{-}\ldots\text{-}E_1) = e_1$. Da mesma forma, como $E_n \leftrightarrow E_{i+1}$ então só pode existir uma instância $e_{i+1} \in E_{i+1}$ tal que $e_{i+1}.(E_{i+1}\text{-}\ldots\text{-}E_n) = e_n$. Como $\{R_i\text{-}E_i, R\text{-}E_{i+1}\}$ é identificador de R_i então existe uma única instância r_i de R_i, onde $r_i.(R_i\text{-}E_i) = e_i$ e $r_i.(R_i\text{-}E_{i+1}) = e_{i+1}$, logo r_i é único.

(2) $R_i \subseteq R$ sss para qualquer $r_i \in R$ então

(i) Existe $r \in R$ tal que $r.(R\text{-}E_1) \equiv r_i.(R_i\text{-}E_i\text{-}\ldots\text{-}E_1)$ e $r.(R\text{-}E_n) \equiv r_i.(R_i\text{-}E_{i+1}\text{-}\ldots\text{-}E_n)$

(ii) r é único

(i) segue direto da definição de R ($R = R_1*\ldots*R_{n-1}[E_1, E_n]$).

Suponha r_i onde $r_i.(R_i\text{-}E_i) = e_i$ e $r_i.(R_i\text{-}E_{i+1}) = e_{i+1}$. Como $E_1 \leftrightarrow E_i$ então só pode existir uma instância $e_1 \in E_1$ tal que $e_1.(E_1\text{-}\ldots\text{-}E_i) = e_i$. Da mesma forma, como $E_n \leftrightarrow E_{i+1}$ então só pode existir uma instância $e_n \in E_n$ tal que $e_n.(E_n\text{-}\ldots\text{-}E_{i+1}) = e_{i+1}$. Como $\{R\text{-}E_1, R\text{-}E_n\}$ é um identificador de R então existe uma única instância r de R, onde $r.(R\text{-}E_1) = e_1$ e $r.(R\text{-}E_n) = e_n$, logo r é único.

De (1) e (2) temos que $R \subseteq R_i$ e $R_i \subseteq R$. □

A seguir, apresentamos dois exemplos de operações de modificação de atributos derivados. No exemplo 6.1 a atualização satisfaz o critério acima e no exemplo 6.2 o critério não é satisfeito.

Exemplo 6.1: *A modificação é realizada em um tipo de relacionamento semanticamente equivalente ao tipo de relacionamento de derivação.*

Considere o esquema local S_1 e o esquema de mediador S_2 da *Figura 6.1*. A *Figura 6.2* mostra os estados iniciais dos tipos ***EMP_M***, R_1 e R_2. Suponha que desejamos modificar o valor do atributo ***#proj_m*** da instância <*José*, *P01*> do tipo ***EMP_M*** atribuindo como novo valor *P02*. Fazendo essa modificação com a modificação do relacionamento correspondente em R_1, temos que o relacionamento em R_1 a ser modificado é aquele que associa *José* ao departamento *D01*. A modificação desse relacionamento é

feita com a modificação do departamento associado a *José* para o departamento que está associado a *P02* (o novo projeto a ser associado a *José*).

Os estados de ***EMP_M*** e ***R_1***, após a modificação de ***R_1***, são mostrados na *Figura 6.3*. Como podemos observar, a instância <*José, D01*> de ***R_1*** foi modificada para <*José, D02*> e a instância <*José, P01*> de ***EMP_M*** foi modificada para <*José, P02*> refletindo a atualização desejada. Como ***R_1*** é semanticamente equivalente ao tipo de relacionamento de derivação de ***#proj_m***, apenas uma instância de ***EMP_M*** foi modificada (existe um mapeamento 1-1 entre ***R_1*** e o tipo de relacionamento de derivação de ***#proj_m***).

Instâncias do tipo ***EMP_M*** **(*emp_m*, *#proj_m*)**
(João, P01)
(Beto, P03)
(José, P01)
(Luís, P02)

Instâncias do tipo ***R_1*** **(*emp*, *#depto*)**
(João, D01)
(Beto, D03)
(José, D01)
(Luís, D02)

Instâncias do tipo ***R_2*** **(*#depto, #proj*)**
(D01, P01)
(D02, P02)
(D03, P03)

Figura 6.2 – Estado inicial dos tipos EMP_M, R_1 e R_2

Instâncias do tipo ***EMP_M*** **(*emp_m*, *#proj_m*)**
(João, P01)
(Beto, P03)
*(José, P02)
(Luís, P02)

Instâncias do tipo ***R_1*** **(*emp*, *#depto*)**
(João, D01)
(Beto, D03)
*(José, D02)
(Luís, D02)

Figura 6.3 – Estado dos tipos EMP_M e R_1 após a modificação em R_1

Exemplo 6.2: *A modificação é realizada em um tipo de relacionamento que não é semanticamente equivalente ao tipo de relacionamento de derivação.*

Suponha que desejamos modificar o valor do atributo ***#proj_m*** da instância <*José, P01*> do tipo ***EMP_M*** atribuindo como novo valor *P04*. Fazendo essa modificação com a

modificação do relacionamento correspondente em $\boldsymbol{R_2}$, temos que o relacionamento em $\boldsymbol{R_2}$ a ser modificado é aquele que associa o departamento *D01* ao projeto *P01*. A modificação desse relacionamento é feita associando o departamento *D01* ao projeto *P04*.

Os estados de ***EMP_M*** e $\boldsymbol{R_2}$, após a modificação de $\boldsymbol{R_2}$, são mostrados na *Figura 6.4*. Como podemos observar, a instância <*D01*, *P01*> de $\boldsymbol{R_2}$ foi modificada para <*D01*, *P04*> e a instância <*José*, *P01*> de ***EMP_M*** foi modificada para <*José*,*P04*> refletindo a atualização desejada. Porém, além da instância <*José*, *P01*> outra instância de ***EMP_M*** foi modificada. A instância <*João*, *P01*> foi modificada para <*João*, *P04*> como conseqüência da mudança do projeto associado ao departamento *D01*. Como $\boldsymbol{R_2}$ não é semanticamente equivalente ao tipo de relacionamento de derivação de ***#proj_m***, a modificação de uma instância em $\boldsymbol{R_2}$ acarretou a modificação de mais de uma instância em ***EMP_M*** (uma instância em $\boldsymbol{R_2}$ está associada a várias instâncias do tipo de relacionamento de derivação de ***#proj_m***).

Instâncias do tipo ***EMP_M*** **(*nger_m*, *#proj_m*)**
*(João, P04)
(Beto, P03)
*(José, P04)
(Luís, P02)

Instâncias do tipo $\boldsymbol{R_2}$ ***(#depto, #proj)***
*(D01, P04)
(D02, P02)
(D03, P03)

Figura 6.4 – Estado dos tipos EMP_M e R_2 após a modificação em R_2

(iii) *Poderá existir no máximo uma ligação com cardinalidade máxima>1 no caminho de derivação do atributo.*

Existem situações onde os critérios (i) e (ii) são satisfeitos, mas mesmo assim não é possível definir o tradutor. Este problema acontece quando existe ambigüidade na hora de determinar a instância do tipo de relacionamento a ser modificada, ou seja existe mais de uma instância que pode ser modificada e produzir o efeito desejado da atualização. Como discutido no Capítulo 5, este tipo de ambigüidade não pode ser resolvida em tempo de definição de mediador e portanto não pode ser tratada pelos nossos algoritmos, uma vez

que os tradutores são gerados na fase de especificação do mediador. A seguir, apresentamos um exemplo que ilustra a situação descrita acima.

Exemplo 6.3:

Considere os esquemas S_4 e S_5 da *Figura 6.5*. e os estados iniciais da *Figura 6.2*. Note que no esquema S_4 tanto R_1 quanto R_2 têm ligações com cardinalidade máxima >1.Suponha que desejamos modificar o valor do atributo ***#proj_m*** da instância <*Luís*, *P02*>do tipo ***EMP_M*** atribuindo como novo valor *P03*. O atributo ***#proj_m*** pode ser modificado quando modifica-se uma instância de R_1 ou uma instância de R_2. Porém, de acordo com o critério (ii), a modificação deve ser realizada em R_1, visto que R_2 não é semanticamente equivalente ao tipo de relacionamento de derivação de ***#proj_m***. Temos que o relacionamento em R_1 a ser modificado é aquele que associa *Luís* ao departamento *D02*. A modificação desse relacionamento deve ser feita com a modificação do departamento associado a *Luís* para o departamento que está associado a *P03* (o novo projeto a ser associado a *Luís*). Como não existe uma correspondência 1-1 entre ***DEPTO*** e ***PROJ***, pode existir mais de uma instância em ***DEPTO*** associada a instância em ***PROJ*** cujo valor de identificador é igual a *P03*. Como conseqüência, a tradução para a atualização é ambígua, ou seja pode existir mais de uma maneira de fazer a modificação em R_1 e realizar a atualização desejada, nesse caso, o tradutor não pode ser definido. Este problema de ambigüidade ocorre quando existe mais de uma ligação com cardinalidade máxima > 1 no caminho de derivação

Quando os critérios (i), (ii) e (iii) são satisfeitos, então garantimos que será possível realizar a atualização em um tipo de relacionamento base semanticamente equivalente ao tipo de relacionamento de derivação do atributo e que não existirá ambigüidade na hora de determinar a instância a ser modificada.

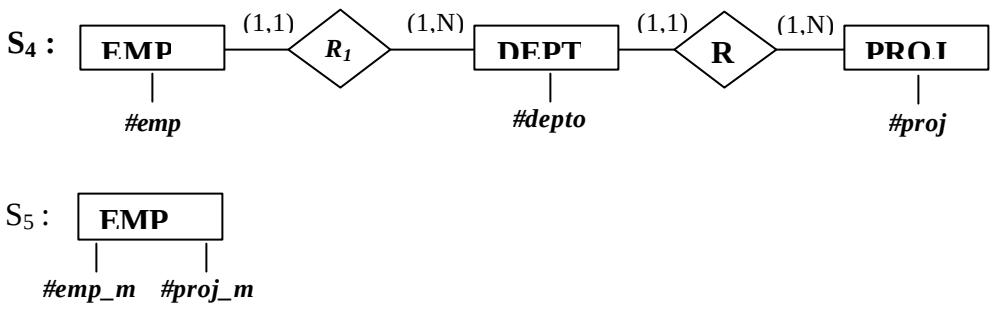


Figura 6.5 – Esquema local S_4 e esquema de mediador S_5

O passo 2.1 do algoritmo *A1*, usa os critérios (i), (ii) e (iii) para determinar o tipo de relacionamento onde deverá ser efetuada a atualização para a manutenção de uma dada assertiva de correspondência de caminhos Ψ, como descrito abaixo:

De acordo com o critério (i), deve-se verificar se o atributo base do atributo derivado é um identificador. Caso seja verdade, então deve-se analisar o número de ligações com cardinalidade máxima > 1 no caminho de derivação. As seguintes possibilidades devem ser consideradas:

- *Existe mais de uma ligação com cardinalidade máxima > 1 (N>1)* : nesse caso, não é possível definir o tradutor (o critério (iii) não é satisfeito).
- *Existe apenas uma ligação com cardinalidade máxima > 1 (N=1)*: nesse caso, a atualização deve ser realizada no tipo de relacionamento que possui a ligação com cardinalidade máxima >1, pois este é o único tipo de relacionamento que é semanticamente equivalente ao tipo de relacionamento de derivação e o tradutor pode ser definido sem ambigüidades.

- *Todas as ligações de relacionamento têm cardinalidade máxima igual a 1 (N=0)* : nesse caso, todos os tipos de relacionamento que fazem parte do caminho de derivação são semanticamente equivalentes ao tipo de relacionamento de derivação e não existe ambigüidade a nível de dados. Assim, o projetista pode escolher qualquer um dos tipos de relacionamento para efetuar

a atualização. É importante lembrar que esta escolha deve ser feita na hora da definição do tradutor.

6.1.2 *Passo 2.2*: Determina a atualização requerida para a manutenção de uma dada assertiva de correspondência de caminho

Escolhido o tipo de relacionamento Y_i, o passo 2.2 determina a atualização requerida em Y_i para a manutenção de uma assertiva de correspondência de caminhos Ψ da forma T_M-$A_M \equiv T_1$-…-T_n-A. No algoritmo *A1* existem três casos que precisam ser tratados separadamente, como explicado abaixo:

- *Caso 1:* $T_i \neq T_1$ e $T_i \neq T_2$

Nesse caso, a ação requerida para Ψ é modificar ou adicionar um relacionamento no tipo de relacionamento T_i escolhido. Para isso, deve-se primeiro obter a instância t_1 de T_1, tal que $t_1 \equiv t_M$ e verificar se t_1 está ligado a alguma instância de T_n através do caminho T_1-…-T_n. No caso dessa condição ser satisfeita, então o atributo A_M já tem um valor definido e a ação requerida é modificar o tipo de relacionamento T_i. Porém, antes de fazer a modificação, deve-se verificar se o valor atual de A é diferente do novo valor a ser atribuído. Se os valores forem diferentes, então devem ser obtidos o relacionamento t_i em T_i onde será feita a modificação e a instância t_{i+1} em T_{i+1} que deverá ser ligada a t_i através da ligação T_i-T_{i+1}. Finalmente, é feita a modificação de t_i, atribuindo t_{i+1} como o novo valor para ligação T_i-T_{i+1} de T_i.

Caso o valor do caminho T_1-…-T_n seja nulo para t_1, então o atributo A_M ainda não tem um valor definido. Nesse caso, a ação requerida é a adição de um relacionamento em T_i. Para isso, devem ser obtidas as instâncias t_{i-1} de T_{i-1} e t_{i+1} de T_{i+1} que comporão o

relacionamento a ser adicionado. Em seguida, procede-se com a adição[1] do relacionamento $<T_i\{<T_{i-1} : t_{i-1}> <T_{i+1} : t_{i+1}>\}>$ em T_i.

No algoritmo *A1*, assim como nos demais algoritmos, quando for solicitada uma instância através de um comando "obtenha" e esta instância não existir no banco de dados, então trataremos esse erro como uma *exceção*. O uso das exceções permite-nos algumas ações corretivas caso os erros ocorram. Nesse caso, a ação corretiva é fazer um *Roolback*, visto que não é possível prosseguir com a atualização.

❖ *Caso 2:* $T_i = T_2$

Nesse caso, a ação requerida para Ψ é modificar ou adicionar um relacionamento no tipo de relacionamento T_2. A única diferença desse caso para o caso anterior é que para fazer a adição em T_2 não precisamos obter a instância de t_{i-1} que comporá o relacionamento. Pois, nesse caso, $T_{i-1} \equiv T_1$, logo a instância t_1 já foi obtida anteriormente.

❖ *Caso 3:* $T_i = T_1$

No caso em que $T_i = T_1$, isso significa que o tipo de relacionamento escolhido é um tipo de relacionamento base de T_M. Nesse caso, a única ação possível é modificar a ligação T_1-T_2 de T_1. Para isso, deve-se obter a instância t_1 de T_1, tal que $t_1 \equiv t_M$ e verificar se o valor do caminho T_1 -…- T_n é nulo para t_1. No caso do caminho não ser nulo, então deve-se fazer a modificação da ligação T_1-T_2, como mostrado anteriormente. Porém, no caso do caminho ser nulo, não será possível fazer a atualização.

6.2 Exemplos de Tradutores para Operações de Modificação de Atributo

[1] Um pedido de adição de relacionamento é expresso por: "Adicione r em R", onde r é uma especificação de R dada por: $<R \{<E_1:e_1> <E_2:e_2>…<E_n:e_n> <A_1:v_1> <A_2:v_2>…<A_k:v_k>\}>$, onde $e_1,…,e_n$ são entidades participantes de r e $v_1,…,v_n$ são os valores para os atributos $A_1,…,A_n$ de r.

A seguir, apresentamos um exemplo que ilustra como gerar o tradutor para uma operação de modificação de atributo usando o algoritmo *A1*:

Considere três bases de dados, cujos esquemas estão apresentados na *Figura 6.6*

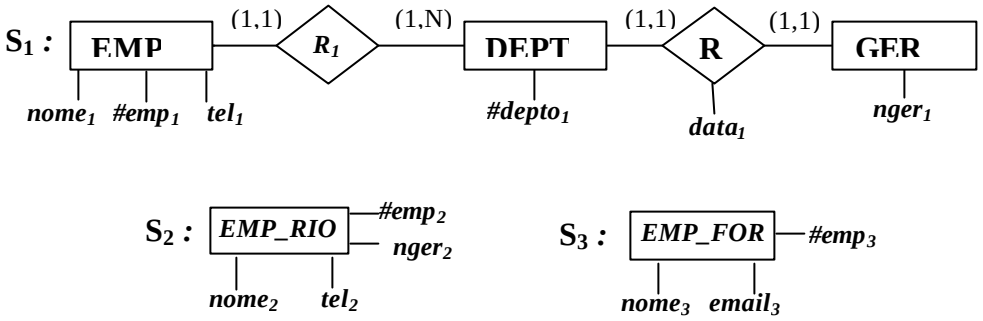


Figura 6.6 - Esquemas S_1, S_2 e S_3

Suponha o mediador chamado ***med,*** que integra as informações em S_1, S_2 e S_3. O esquema de ***med*** é mostrado na *Figura 6.7*.

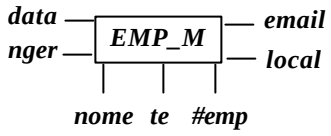


*Figura 6.7 - Esquema de **med***

Integrando o esquema do mediador com os esquemas dos bancos de dados locais S_1, S_2 e S_3, obtemos as seguintes assertivas de correspondência:

Assertivas de Correspondência de Tipos:

AC_1: ***EMP_M*** ≡ ***EMP***

AC_2: ***EMP_M*** ≡ Gen(***EMP_RIO*** , ***EMP_FOR***)

AC_3: ***EMP_RIO*** ≡ ***EMP_M*** [***local*** = ‘rio’]

AC_4: ***EMP_FOR*** ≡ ***EMP_M***[***local*** = ‘for’]

Assertivas de Correspondência de Atributos:

AC_5: ***nome*** ≡ ***nome₁***

AC_6: ***nome*** ≡ ***nome₂***

AC_{10}: ***#emp*** ≡ ***#emp₁***

AC_{11}: ***#emp*** ≡ ***#emp₂***

AC_8: ***tel*** ≡ ***tel₁***

AC_9: ***tel*** ≡ ***tel₂***

AC_{13}: ***local*** ≡ [***EMP_RIO*** :'rio';***EMP_FOR*** :'for']

AC_{14}: ***email*** ≡ ***email₃***

AC_{15}: ***nger*** ≡ ***nger₂***

Assertivas de Correspondência de Caminhos:

AC_{16}: ***EMP_M-nger*** ≡ ***EMP-R₁-DEPTO-R₂-GER-nger₁***

AC_{17}: ***EMP_M-data*** ≡ ***EMP-R₁-DEPTO-R₂-data₁***

O tradutor para a operação de modificação do atributo ***nome***, denominado ***Modifica_nome*** é mostrado na *Figura 6.8*. As assertivas de correspondência relevantes para esta operação e que portanto foram consideradas pelo algoritmo *A1* são as assertivas AC_5, AC_6 e AC_7. As ações requeridas, geradas pelo algoritmo *A1*, para cada uma dessas assertivas são apresentadas no tradutor ***Modifica_nome***. A ação requerida para a assertiva AC_5 consiste em determinar a instância e_1 do tipo ***EMP*** que é semanticamente equivalente a e_M (a entidade de mediador que estamos modificando) e então, modificar o valor do atributo ***nome₁*** de e_1 de acordo com o valor atribuído a ***nome***. As ações requeridas para as assertivas AC_6 e AC_7 são semelhantes a esta da assertiva AC_5.

Modifica_nome(e_M, ***v***)

{

/ Ação requerida para a assertiva de correspondência AC_5 */*

Se existir uma instância e_1 em ***EMP*** tal que $e_1 \equiv e_M$ *então*

e₁.nome₁ : =‘***v***’;

/ Ação requerida para a assertiva de correspondência AC_6 */*

Se existir uma instância ***e₂*** em ***EMP_RIO*** tal que ***e₂*** ≡ ***e_M*** *então*
e₂.nome₂ : =‘***v***’;

/ Ação requerida para a assertiva de correspondência AC_7 */*

Se existir uma instância ***e₃*** em ***EMP_FOR*** tal que ***e₃*** ≡ ***e_M*** *então*
e₃.nome₃ : =‘***v***’;

}

Figura 6.8 – Tradutor para a operação de modificação do atributo ***nome***

O tradutor para a modificação do atributo ***nger***, denominado ***Modifica_nger***, é mostrado na *Figura 6.9*. As assertivas de correspondência relevantes para esta operação e que portanto foram consideradas pelo algoritmo *A1* são as assertivas AC_{15} e AC_{16}. As ações requeridas para cada uma dessas assertivas estão descritas a seguir:

- AC_{15} : a ação requerida para esta assertiva é determinar a instância do tipo ***EMP_RIO*** que é semanticamente equivalente a ***e_M*** e então, modificar o valor do atributo ***nger₂*** de acordo com o valor atribuído a ***nger***.

- AC_{16}: esta é uma assertiva de correspondência de caminho, que é tratada pelo passo 2 do algoritmo *A1*. Como nesse caso apenas ***R₁*** é funcionalmente equivalente ao tipo de relacionamento de derivação de ***nger***, então ***R₁*** é o tipo de relacionamento onde deve ser efetuada a modificação. No caso de ***e₁*** (***e₁*** ≡ ***e_M***) ter um valor definido para ***nger₁***, a atualização consiste em modificar o departamento que está ligado a ***e₁***, atribuindo como novo departamento a instância ***d*** de ***DEPTO*** que está ligada a instância ***g*** de ***GER*** cujo valor de ***nger₁*** é ‘***v***’. No caso de ***nger₁*** ainda não ter um valor definido, devemos adicionar um relacionamento em ***R₁*** ligando ***e₁*** com a instância ***d*** de ***DEPTO***, que está ligada a instância ***g*** de ***GER*** cujo valor de ***nger₁*** é ‘***v***’.

Tradutor ***Modifica_nger*** (***e_M***, ***v***)
{
/ Ação requerida para a assertiva AC_{15} */*

Se existir uma instância e_2 de ***EMP_RIO*** tal que $e_2 \equiv e_M$ *então*
$e_2.nger_2$: ='**v**' ;

/ Ação requerida para a assertiva AC_{16} */*

Obtenha e_1 em ***EMP*** tal que $e_1 \equiv e_M$;
Se $e_1.(EMP\text{-}R_1\text{-}DEPTO\text{-}R_2\text{-}GER) \neq$ 'nulo' *então*
/ Faz a modificação em Y_i */*
Seja ***g*** a instância de ***GER*** tal que $e_1.(EMP\text{-}R_1\text{-}DEPTO\text{-}R_2\text{-}GER) = g$;
Se $g.nger \neq$ '**v**' *então*
Obtenha r_1 em R_1 tal que $e_1.(EMP\text{-}R_1) = r_1$;
Obtenha ***d*** em ***DEPTO*** tal que $d.(DEPTO\text{-}R_2\text{-}GER\text{-}nger_1)$ = '**v**';
$r.(R_1\text{-}DEPTO)$: = ***d***;
senão
/ Faz a adição em Y_i. */*
Obtenha ***d*** em ***DEPTO*** tal que $d.(DEPTO\text{-}R_2\text{-}GER\text{-}nger_1)$ = '**v**';
Adicione R_1 {< ***EMP*** : e_1 > <***DEPTO*** : ***d*** >} *em* R_1;
}

Figura 6.9 - Tradutor para a operação de modificação do atributo ***nger***

CAPÍTULO 7

ATUALIZAÇÃO DE TIPOS DE ENTIDADE

Neste capítulo, descrevemos os algoritmos que geram tradutores para as operações básicas de atualização de tipos de entidade de mediador, as quais são:

- *Adição* de uma instância em um Tipo de Entidade
- *Remoção* de uma instância de um Tipo de Entidade

Na seção 7.1, descrevemos o algoritmo que gera tradutores para as operações de *adição*. Nesta seção também descrevemos algoritmos auxiliares que nós desenvolvemos para gerar os Construtores dos tipos de entidade, os quais são utilizados para criar as instâncias que serão adicionadas nos tipos base. No final da seção 7.1, apresentamos alguns exemplos de tradutores para operações de *adição* em tipos de entidade. Na seção 7.2, descrevemos o algoritmo que gera tradutores para as operações de *remoção*. No final desta seção, mostramos alguns exemplos de tradutores para operações de *remoção* em tipos de entidade.

7.1 Definindo Tradutores para as Operações de Adição em Tipos de Entidade

O algoritmo *E1*, apresentado na seção A.2, gera tradutores para as operações de *adição* em tipos de entidade de mediador. Neste trabalho, um pedido de *adição* de entidade é expresso por "*Adicione* ***e*** *em* ***E*** ", onde ***e*** é uma especificação de uma instância do tipo de entidade ***E*** dada por < ***E*** { <A_1:v_1> <A_2:v_2>…<A_n:v_n>}>, onde $A_1 \ldots A_n$ são atributos de ***E*** e $v_1,\ldots,v_n$ são os valores atribuídos a estes atributos. O tradutor para a operação de *adição* em um tipo de entidade E_M é denominado ***Adiciona_***E_M e o parâmetro passado para este tradutor é a especificação da entidade que desejamos adicionar.

As assertivas de correspondência relevantes para a operação de *adição* em um tipo de entidade de mediador e que portanto devem ser consideradas no algoritmo *E1* são:

- *Assertivas de Correspondência de Tipos*
- *Assertivas de Correspondência de Caminhos*
- *Assertivas de Dependência Existencial*

As assertivas de dependência existencial consideradas no algoritmo *E1* são apenas as assertivas que especificam restrições referenciais, os demais tipos não são tratados. O algoritmo *E1* é dividido em três passos: o passo 1 define as atualizações requeridas para a manutenção das assertivas de correspondência de tipos, o passo 2 define as atualizações requeridas para a manutenção das assertivas de dependência existencial e o passo 3 define as atualizações requeridas para a manutenção das assertivas de correspondência de caminhos

❖ *Manutenção das Assertivas de Correspondência de Tipos*

O passo 1 do algoritmo *E1* determina as atualizações requeridas para a manutenção das assertivas de correspondência de tipos. A Tabela 1 do Apêndice B define as ações requeridas para a manutenção das assertivas de correspondência de tipos referentes à *adição* de uma entidade $\boldsymbol{e_M}$ em um tipo de entidade $\boldsymbol{E_M}$. A primeira assertiva de correspondência de tipo tratada pelo algoritmo *E1* é a assertiva de equivalência $\boldsymbol{E_M \equiv E}$, onde $\boldsymbol{E_M}$ é um tipo de entidade de mediador e $\boldsymbol{E}$ é um tipo base. Para esta assertiva, como indicado na Tabela 1, a ação requerida para um pedido de atualização "*Adicione* $\boldsymbol{e_M}$ *em* $\boldsymbol{E_M}$" é adicionar uma instância $\boldsymbol{e}$ em $\boldsymbol{E}$ tal que $\boldsymbol{e}$ é semanticamente equivalente a $\boldsymbol{e_M}$. Antes de fazer a adição de $\boldsymbol{e}$ em $\boldsymbol{E}$ é preciso, primeiro, criar a instância $\boldsymbol{e}$ a partir de $\boldsymbol{e_M}$. Como mostrado no algoritmo *E1*, usamos o construtor *cria_*$\boldsymbol{E}$*_de_*$\boldsymbol{E_M}$ para criar uma instância de $\boldsymbol{E}$ a partir de uma instância de $\boldsymbol{E_M}$. Este construtor recebe como entrada $\boldsymbol{e_M}$ e retorna uma instância $\boldsymbol{e}$ de $\boldsymbol{E}$, tal que $\boldsymbol{e \equiv e_M}$.

Para cada um dos tipos base de um tipo de mediador deve existir um construtor que é usado para criar uma instância do tipo base a partir de uma instância do tipo de mediador. O nome dos construtores será expresso por *cria<tipo base>_de_<tipo de mediador>*. Na seção 7.1.1, serão descritos os algoritmos *C1* e *C2*, que são utilizados para gerar os construtores requisitados pelos tradutores de operações de *adição* de tipos de entidade. É importante lembrar que um tradutor só poderá gerar corretamente uma tradução se todos os construtores utilizados por este tradutor já tiverem sido gerados.

As ações requeridas para a manutenção das demais assertivas de correspondência de tipos, tratadas no passo 1 do algoritmo *E1*, são estas definidas na Tabela 1 do Apêndice B.

❖ *Manutenção das Assertivas de Dependência Existencial*

O passo 2 do algoritmo *E1* define as atualizações requeridas para a manutenção das assertivas de dependência existencial. A Tabela 3 do Apêndice B define as ações requeridas

para a manutenção das assertivas de dependência existencial referentes à *adição* de uma instância e_M em um tipo E_M. Como dito inicialmente, as assertivas consideradas pelo algoritmo *E1* são aquelas que especificam uma restrição referencial, que pode ser tanto parcial como total.

❖ *Manutenção das Assertivas de Correspondência de Caminhos*

O passo 3 do algoritmo *E1* define as atualizações requeridas para a manutenção das assertivas de correspondência de caminhos. Uma assertiva de correspondência de caminhos da forma E_M-$A_M \equiv Y_1$-Y_2-Y_3…-Y_n-A, onde $Y_1,\ldots,Y_n$ são tipos, especifica o caminho de derivação do atributo A_M. Como visto no capítulo 6, no caso de ser atribuído um valor para o atributo derivado A_M, deve-se adicionar ou modificar um relacionamento em um dos tipos de relacionamento que fazem parte do caminho de derivação de A_M. O procedimento para definir as atualizações requeridas para a manutenção das assertivas de correspondência de caminhos, na *adição* em um tipo de entidade, é o mesmo adotado para a operação de *modificação* de atributos, o qual já foi explicado no capítulo 6. Entretanto, nos casos em que Y_1 é um tipo de relacionamento, ele não poderá ser o tipo escolhido para a atualização. Além disso, as assertivas de correspondência de caminhos da forma E_M-$A_M \equiv Y_1$-Y_2-A, onde Y_1 é um tipo de relacionamento e A é um atributo de Y_2, não precisam ser consideradas neste passo, uma vez que elas são usadas no passo 1 para definir as ligações de entidade de Y_1.

7.1.1 Construtores para Tipos de Entidade de Mediador

Como visto anteriormente, a adição de uma instância em um tipo de entidade de mediador requer a adição de instâncias nos tipos base correspondentes. Para fazer a adição nos tipos base, é preciso, primeiro, criar as instâncias a serem adicionadas. Como estas instâncias devem ser semanticamente equivalentes a entidade de mediador que está sendo adicionada, devemos criá-las a partir desta entidade de mediador. Na nossa abordagem,

propomos que as instâncias dos tipos base sejam criadas através de construtores que recebem como entrada uma entidade de mediador e retornam uma instância de um determinado tipo base. É válido lembrar que estes construtores podem ser gerados automaticamente através de algoritmos.

Como os tipos base de um tipo de entidade, podem ser tanto tipos de entidade como tipos de relacionamento, temos que a *adição* em um tipo de entidade pode ocasionar tanto a adição de uma instância em um tipo de relacionamento base como em um tipo de entidade base. Sendo assim, neste caso, dois tipos de construtores são necessários:

- *Construtor de entidades* : recebe uma entidade $\boldsymbol{e_M}$ de um tipo de entidade de mediador $\boldsymbol{E_M}$ e retorna uma entidade $\boldsymbol{e}$ de um tipo de entidade $\boldsymbol{E}$, tal que $\boldsymbol{e} \equiv \boldsymbol{e_M}$.

- *Construtor de relacionamentos*: recebe uma entidade $\boldsymbol{e_M}$ de um tipo de entidade de mediador $\boldsymbol{E_M}$ e retorna um relacionamento $\boldsymbol{r}$ de um tipo de relacionamento $\boldsymbol{R}$, tal que $\boldsymbol{r} \equiv \boldsymbol{e_M}$

Para gerar estes construtores desenvolvemos dois algoritmos: o algoritmo *C1*, que é utilizado para gerar os construtores de entidades e o algoritmo *C2*, que é utilizado para gerar os construtores de relacionamentos.

O algoritmo *C1*, apresentado na seção A.4 do Apêndice A, gera o construtor para criar uma entidade $\boldsymbol{e}$ de um tipo base $\boldsymbol{E}$ a partir de uma especificação de entidade $\boldsymbol{e_M} = < \boldsymbol{E_M}\{ <\boldsymbol{A_1} :\boldsymbol{v_1}> <\boldsymbol{A_2} :\boldsymbol{v_2}>\ldots<\boldsymbol{A_n} :\boldsymbol{v_n}>\}>$, onde $\boldsymbol{E_M}$ é um tipo de entidade de mediador . Para criar uma entidade $\boldsymbol{e}$ a partir de $\boldsymbol{e_M}$, deve-se determinar valores para os atributos de $\boldsymbol{e}$ a partir de $\boldsymbol{e_M}$. Como mostrado no algoritmo *C1*, os valores dos atributos de $\boldsymbol{e}$ são determinados a partir das assertivas de correspondência de atributo do mediador.

O algoritmo *C2*, apresentado na seção A.4 do Apêndice A, gera o construtor para criar um relacionamento ***r*** de um tipo base ***R*** a partir de uma especificação de entidade e_M = < E_M\{ <A_1 :v_1> <A_2 :v_2>…<A_n :v_n>}>, onde E_M é um tipo de entidade de mediador. Para criar um relacionamento ***r*** a partir de e_M, deve-se determinar as entidades participantes ***r*** e os valores para os atributos de ***r*** a partir de e_M. Como mostrado no algoritmo *C2*, as entidades participantes de ***r*** e os valores para os atributos de ***r*** são determinados a partir de assertivas de correspondência de caminhos e de atributos do mediador.

7.1.2 Exemplos de Tradutores para Operações de Adição em Tipos de Entidade de Mediador

A seguir, apresentamos alguns exemplos de tradutores para operações de adição em tipos de entidade, que foram gerados usando o algoritmo *E1*:

Considere duas bases de dados com esquemas S_1 e S_2 , apresentados na *Figura 7.1* .

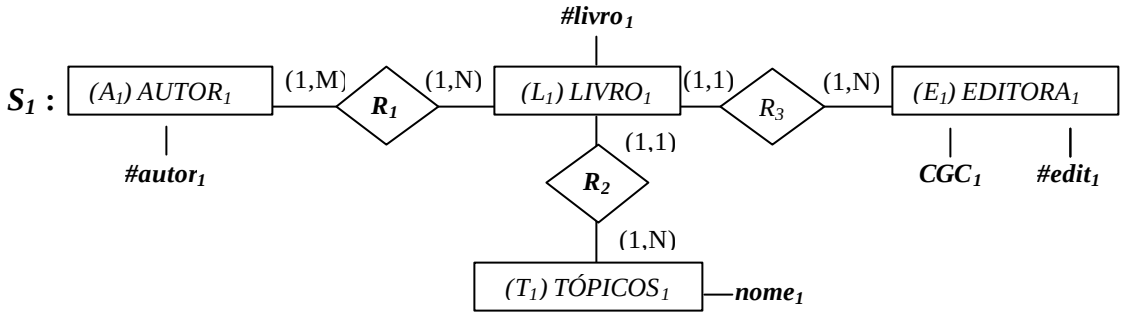


Figura 7.1 - Esquemas S_1 e S_2

Suponha o mediador chamado ***med₁,*** que integra as informações em ***S₁*** e ***S₂***. O esquema de ***med₁*** é mostrado na *Figura 7.2*.

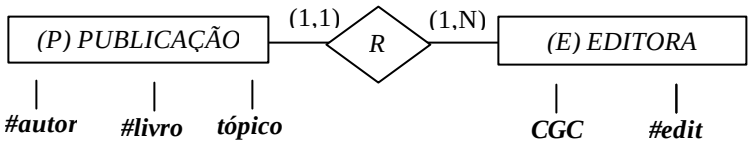


Figura 7.2 - Esquema de ***med₁***

Integrando o esquema do mediador com os esquemas dos bancos de dados locais ***S₁*** e ***S₂***, obtemos as seguintes assertivas de correspondência:

Assertivas de Correspondência de Tipos:

$$AC_1: P \circ R_1$$

AC_2: $E \equiv E_1$

AC_3: $L_2 \equiv P$ [***tópico*** = ‘informática’]

Assertivas de Correspondência de Atributos:

AC_4: $\#edit \equiv \#edit_1$ AC_6: $\#autor \equiv \#autor_2$

AC_5: $CGC \equiv CGC_1$ AC_7: $\#livro \equiv \#livro_2$

Assertivas de Correspondência de Caminhos:

AC_8: $P - \#autor \equiv R_1\text{-}A_1\text{-}\#autor_1$

AC_9: $P - \#livro \equiv R_1 - L_1 - \#livro_1$

AC_{10}: P - *tópico* $\circ$ $R_1 - L_1 - R_2 - T_1 - nome_1$

Assertivas de Dependência Existencial:

AC_{11}: $R[R\text{-}P\text{-}\#livro, R\text{-}E\text{-}\#edit] \equiv R_3[R_3 - L_1 - \#livro_1, R_3\text{-}E_1\text{-}\#edit_1]$

AC_{12}: P [$P\text{-}\#autor, P\text{-}\#livro$] $\equiv R_1[R_1\text{-}A_1\text{-}\#autor_1, R_1 - L_1 - \#livro_1]$

O tradutor para a operação de adição em ***EDITORA*,** denominado ***Adiciona_EDITORA***, é mostrado na *Figura 7.3*. A única assertiva de correspondência relevante para esta operação, e que portanto foi considerada pelo algoritmo *E1*, é a assertiva AC_2. A ação requerida para a manutenção de AC_2 é criar uma instância e_1 de $EDITORA_1$, tal que $e_1 \equiv e_M$, e em seguida, adicionar e_1 em $EDITORA_1$. Para criar e_1 usamos o construtor *cria_**$EDITORA_1$**_de_**EDITORA***, apresentado na *Figura 7.4*. Este construtor é gerado usando o algoritmo *C1*, onde os valores dos atributos de e_1 são determinados pelas assertivas AC_4 e AC_5.

Adiciona_ EDITORA **(**e_M**)**

{

/ Ação requerida para a assertiva de correspondência* AC_2 **/*

e_1 : = *cria_**$EDITORA_1$**_de_**EDITORA*** **(**e_M**)**;

Adicione e_1 *em* ***$EDITORA_1$***;

}

Figura 7.3 - Tradutor para a operação de adição em ***EDITORA***.

---

*cria_**$EDITORA_1$**_de_**EDITORA*** **(**e_M**)**

{

/ e_M = < ***EDITORA***{ <***#edit*** :v_1> <***CGC*** :v_2>}>, *onde* v_1 *e* v_2 *são variáveis cujos valores serão passados como parâmetros em tempo de atualização */*

$v_{\#edit_1}$: = v_1;

v_{CGC_1} : = v_2;

Retorne (< ***$EDITORA_1$***{ <***$\#edit_1$*** : $v_{\#edit_1}$ > <***CGC_1*** : v_{CGC_1} >}>);

}

Figura 7.4 – Construtor utilizado pelo Tradutor ***Adiciona_EDITORA***

O tradutor para a operação de adição em ***PUBLICAÇÃO,*** denominado ***Adiciona_PUBLICAÇÃO,*** é mostrado na *Figura 7.5*. As assertivas de correspondência relevantes para esta operação, e que portanto foram consideradas pelo algoritmo *E1*, são as assertivas AC_1, AC_3 e AC_{10}. As ações requeridas para estas assertivas estão descritas logo abaixo:

- AC_1 (***P*** ≡ $\boldsymbol{R_1}$) : a ação requerida para AC_1 é criar uma instância $\boldsymbol{r_1}$ de $\boldsymbol{R_1}$, tal que $\boldsymbol{r_1} \equiv \boldsymbol{e_M}$, e em seguida, adicionar $\boldsymbol{r_1}$ em $\boldsymbol{R_1}$. Para criar $\boldsymbol{r_1}$ usamos o construtor *cria_**R**$_1$_de_**PUBLICAÇÃO,*** apresentado na *Figura 7.6*. Este construtor é gerado pelo algoritmo *C2*, onde as entidades participantes de $\boldsymbol{r_1}$ são determinadas pelas assertivas AC_8 e AC_9.

- AC_3 (***L***$_2$ ≡ ***P*** [***tópico*** = ‘informática’]): a ação requerida para esta assertiva é criar uma instância ***l*** de ***LIVROS_DE_INFORMÁTICA***, tal que ***l*** ≡ $\boldsymbol{e_M}$, e em seguida, adicionar ***l*** em ***LIVROS_ DE_INFORMÁTICA***. Para criar ***l*** usamos o construtor *cria_**LIVROS_DE_INFORMÁTICA**_de_**PUBLICAÇÃO,*** apresentado na *Figura 7.7*. Este construtor é gerado pelo algoritmo *C1*, onde os valores dos atributos de ***l*** são determinados pelas assertivas AC_6 e AC_7.

- AC_{10} (***P-tópico*** ≡ $\boldsymbol{R_1}$-$\boldsymbol{L_1}$-$\boldsymbol{R_2}$-$\boldsymbol{T_1}$-$\boldsymbol{nome_1}$) : a ação requerida para esta assertiva é modificar ou adicionar um relacionamento em um dos tipos de relacionamento do caminho de derivação do atributo ***tópicos***. O caminho de derivação de tópicos tem apenas dois tipos de relacionamentos: $\boldsymbol{R_1}$ e $\boldsymbol{R_2}$. Como $\boldsymbol{R_1}$ é o tipo de relacionamento base de ***PUBLICAÇÃO,*** a atualização não pode ser realizada em $\boldsymbol{R_1}$. Logo, a ação requerida é modificar ou adicionar um relacionamento em $\boldsymbol{R_2}$, como indicado na *Figura 7.5*.

Adiciona_ PUBLICAÇÃO (e_M)

{

/ Ação requerida para a assertiva de correspondência AC_1 */*

r_1 : = *cria* $_R_1_$de$_$***PUBLICAÇÃO***(e_M);

Adicione r_1 *em* R_1;

/ Ação requerida para a assertiva de correspondência AC_3 */*

Se e_M***.tópico*** = 'informática' *então*

l : = *cria_* ***LIVROS_DE_INFORMÁTICA***_*de*_***PUBLICAÇÃO*** (e_M);

Adicione l *em* ***LIVROS_DE_INFORMÁTICA***;

/ Ação requerida para a assertiva de correspondência AC_{10} */*

Obtenha r_1 de R_1 tal que $r_1 \equiv e_M$

Se $r_1.(R_1 - L_1 - R_2 - T_1 - nome_1) \neq$ 'nulo' *então*

Seja t_1 a instância de T_1 tal que $r_1.(R_1 - L_1 - R_2 - T_1) = t_1$

Se $t_1.nome_1 \neq e_M.tópico$ *então*

Obtenha r_2 de R_2 tal que $r_1.(R_1 - L_1 - R_2) = r_2$;

Obtenha t de T_1 tal que $t.(T_1 - nome_1) = e_M.tópico$;

$r_2.(R_2 - T_1)$: = t ;

senão

Obtenha l de L_1 tal que $r_1.(R_1 - L_1) = l$;

Obtenha t de ***TÓPICOS*** tal que $t.(T_1 - nome_1) = e_M.tópico$;

Adicione $<R_2$ {< L : l> <T : t >}> em R_2;

}

Figura 7.5- Tradutor para a operação de adição em ***PUBLICAÇÃO***

cria$\_R_1\_$*de*$\_$***PUBLICAÇÃO*** (e_M)

{

*/** e_M = <***PUBLICAÇÃO*** *{<***#autor***:*v_1*> <***#livro***:*v_2*> <***tópico***:*v_3*>}>, onde v_1, v_2 e v_3 são variáveis cujos valores serão passados como parâmetros em tempo de atualização */*

Obtenha a de ***AUTOR***, tal que $a.\#autor_1 = v_1$;

Obtenha l de ***LIVRO***, tal que $l.\#livro_1 = v_2$;

Retorne (< R_1{<***AUTOR*** : a> <***LIVRO***: l> }>);

}

Figura 7.6- Construtor cria $\_R_1\_$*de*$\_$***PUBLICAÇÃO***

*cria_**LIVROS_DE_INFORMÁTICA**_de_**PUBLICAÇÃO*** (***e*****$_M$**)

{

/* ***e*****$_M$** = <***PUBLICAÇÃO*** {<***#autor*** :***v*****$_1$**> <***#livro*** :***v*****$_2$**>}>, *onde* v_1 *e* v_2 *são variáveis cujos valores serão passados como parâmetros em tempo de atualização* */

v**$_{\#autor_2}$** : = ***v*****$_1$**;

v**$_{\#livro_2}$** : = ***v*****$_2$**;

v**$_{\#edit_2}$** : = 'nulo'

Retorne (<***LIVROS_DE_INFORMÁTICA***{<***#autor*****$_2$**:***v*****$_{\#autor_2}$** > <#***livro*****$_2$**:***v*****$_{\#livro_2}$** >

<***#edit*****$_2$**:'nulo'>}>);

}

Figura 7.7- Construtor ***cria _LIVROS_DE_INFORMÁTICA_de_PUBLICAÇÃO***

7.2 Definindo Tradutores para as Operações de Remoção em Tipos de Entidade

O algoritmo *E2*, apresentado na seção A.2, gera tradutores para as operações de remoção em tipos de entidade de mediador. Neste trabalho, um pedido de remoção de entidade é expresso por: *"Remova **e** de **E** "*, onde ***e*** é uma entidade do tipo de entidade ***E***. Denominaremos de ***Remove_E*****$_M$**, o tradutor para a operação de remoção em um tipo de entidade ***E*****$_M$**. O único parâmetro passado para este tradutor é a entidade ***e*****$_M$** que desejamos remover de ***E*****$_M$**.

As assertivas de correspondência relevantes para a operação de *remoção* e que portanto devem ser consideradas no algoritmo *E2* são:

- *Assertivas de Correspondência de Tipos*
- *Assertivas de Dependência Existencial*

O algoritmo *E2* é dividido em dois passos: o passo 1 define as atualizações requeridas para a manutenção das assertivas de correspondência de tipos e o passo 2 define as atualizações requeridas para a manutenção das assertivas de dependência existencial.

❖ *Manutenção das Assertivas de Correspondência de Tipos*

O passo 1 do algoritmo *E2* define as atualizações requeridas para a manutenção das assertivas de correspondência de tipos. A Tabela 2 do Apêndice B define as ações requeridas para a manutenção das assertivas de correspondência de tipo referentes a *remoção* em um tipo de entidade. Semelhante ao que ocorre com a operação de *adição,* a remoção de uma entidade pode ocasionar tanto a remoção de entidades quanto a remoção de relacionamentos nos tipos base correspondentes. Quando a ação requerida, para preservar uma dada assertiva de correspondência de tipo, consiste em remover uma instância de um dos tipos base, deve existir uma instância deste tipo base que seja semanticamente equivalente à entidade de mediador a ser removida. Como o tipo base e o tipo de entidade de mediador são semanticamente equivalentes, então sempre existirá um mapeamento 1-1 entre as suas instâncias, que pode ser usado para determinar qual a instância do tipo base que deve ser removida. Este mapeamento é estabelecido através de assertivas de correspondência de caminhos, que estabelecem o mapeamento entre a chave do tipo base e a chave do tipo de entidade de mediador.

❖ *Manutenção das Assertivas de Dependência Existencial*

O passo 2 do algoritmo *E2* define as atualizações requeridas para a manutenção das assertivas de DE. A Tabela 4 do Apêndice B define as ações requeridas para a manutenção das assertivas de dependência existencial referentes à *remoção* de uma instância $\boldsymbol{e_M}$ em um tipo $\boldsymbol{E_M}$. Assim como no algoritmo *E1*, as assertivas de DE consideradas pelo algoritmo *E2* são aquelas que especificam uma restrição referencial, que pode ser tanto parcial como total.

7.2.1 Exemplos de Tradutores para Operações de Remoção em Tipos de Entidade de Mediador

A seguir, apresentamos alguns exemplos de tradutores para operações de *remoção* em tipos de entidade, que foram gerados usando o algoritmo *E2*.

Considere os esquemas S_1 e S_2, apresentados na *Figura 7.1*, o esquema do mediador med_1 apresentado na *Figura 7.2* e as assertivas de correspondência obtidas da integração do esquema de med_1 com os esquemas S_1 e S_2, apresentadas na seção 7.1.2.

- O tradutor para a operação de *remoção* em ***PUBLICAÇÃO***, denominado ***Remove_PUBLICAÇÃO***, é mostrado na *Figura 7.8*. As assertivas de correspondência relevantes para esta operação, e que portanto foram consideradas pelo algoritmo *E2*, são as assertivas AC_1 e AC_3.

Remove_ PUBLICAÇÃO (e_M)

{

/ Ação requerida para a assertiva de correspondência* AC_1 **/*

Se existir r_1 em R_1 tal que $r_1 \equiv e_M$ *então*

Remova r_1 *de* R_1 ;

/ Ação requerida para a assertiva de correspondência* AC_3 **/*

Se existir ***l*** em ***LIVROS_DE_INFORMÁTICA*** tal que $l \equiv e_M$ *então*

Remova ***e*** *de* ***E***;

}

Figura 7.8- Tradutor para a operação de remoção no tipo ***PUBLICAÇÃO***

O tradutor para a operação de *remoção* no tipo de entidade ***EDITORA***, denominado ***Remove_EDITORA***, é mostrado na *Figura 7.9*. A única assertiva de correspondência

relevante para esta operação, e que portanto foi considerada pelo algoritmo *E2*, é a assertiva AC_2.

***Remove_ EDITORA*(e_M)**
{
/ Ação requerida para a assertiva de correspondência AC_2 */*

Se existir e_1 em $EDITORA_1$ tal que $e_1 \equiv e_M$ *então*

Remova e_1 de $EDITORA_1$;
}

Figura 7.9- Tradutor para a operação de remoção no tipo ***EDITORA***

CAPÍTULO 8

ATUALIZAÇÃO DE TIPOS DE RELACIONAMENTO

Neste capítulo descrevemos os algoritmos que geram tradutores para as operações básicas de atualização de tipos de relacionamento de mediador, as quais são:

- *Adição* de uma instância em um Tipo de Relacionamento
- *Remoção* de uma instância de um Tipo de Relacionamento

Na seção 8.1, descrevemos o algoritmo que gera tradutores para as operações de *adição*. Nesta seção, também apresentamos os algoritmos que nós desenvolvemos para gerar construtores, que criam entidades e relacionamentos de tipos base a partir de um relacionamento de um tipo de mediador. No final desta seção, apresentamos alguns exemplos de tradutores para operações de *adição* em tipos de relacionamento. Na seção 8.2, descrevemos o algoritmo que gera tradutores para as operações de *remoção*. No final desta seção apresentamos exemplos de tradutores para operações de *remoção* de tipos de relacionamento.

8.1 Definindo Tradutores para as Operações de Adição em Tipos de Relacionamento

O algoritmo *R1*, apresentado na seção A.3 do Apêndice A, gera tradutores para as operações de *adição* em tipos de relacionamento. Neste trabalho, um pedido de *adição* de relacionamento é expresso por *"Adicione* ***r*** *em* ***R*** *"*, onde ***r*** é uma especificação de ***R*** dada por <***R*** {<E_1:e_1> <E_2:e_2>…<E_n:e_n> <A_1:v_1> <A_2:v_2>…<A_k:v_k>}>, onde $E_1,\ldots,E_n$ são tipos de entidades participantes de ***R*** e $e_1,\ldots,e_n$ são instâncias destes tipos, $A_1,\ldots,A_n$ são atributos de ***R*** e $v_1,\ldots,v_n$ são os valores destes atributos. O tradutor para a operação de adição em um tipo de relacionamento R_M é denominado ***Adiciona_***R_M e o parâmetro passado para este tradutor é a especificação do relacionamento que desejamos adicionar.

As assertivas de correspondência relevantes para a operação de *adição* em um tipo de relacionamento e que portanto devem ser consideradas no algoritmo *R1* são:

- *Assertivas de Correspondência de Tipos*
- *Assertivas de Dependência Existencial*
- *Assertivas de Correspondência de Caminhos*

As assertivas de dependência existencial consideradas no algoritmo *R1* são apenas as assertivas que especificam restrições referenciais, os demais tipos não são tratados. O algoritmo *R1* é dividido em três passos: o passo 1 define as atualizações requeridas para a manutenção das assertivas de correspondência de tipos, o passo 2 define as atualizações requeridas para a manutenção das assertivas de dependência existencial e o passo 3 define as atualizações requeridas para a manutenção das assertivas de correspondência de caminho.

❖ *Manutenção das Assertivas de Correspondência de Tipos*

O passo 1 do algoritmo *R1* determina as atualizações requeridas para a manutenção das assertivas de correspondência de tipos. Como as assertivas de correspondência de tipos são as mesmas para tipos de entidade e tipos de relacionamento, temos que a ação requerida para a manutenção destas assertivas na adição em um tipo de relacionamento são apresentadas na tabela 1 do Apêndice B. Da mesma maneira que ocorre com a adição em tipos de entidade, a adição em um tipo de relacionamento requer a adição de instância nos tipos base. Neste caso, a instância a ser adicionada no tipo base deve ser semanticamente equivalente ao relacionamento de mediador que desejamos adicionar. Para criar estas instâncias, utilizaremos construtores que recebem um relacionamento de mediador e retornam uma instância semanticamente equivalente a este relacionamento.

Como mostrado no algoritmo *R1*, usamos o construtor *cria_**E**_de_**R_M*** para criar uma instância de ***E*** a partir de uma instância de ***R_M***. Este construtor recebe como entrada um relacionamento ***r_M*** de um tipo de mediador ***R_M*** e retorna uma instância ***e*** de ***E***, tal que ***$e \equiv r_M$***. É importante lembrar que um tradutor só poderá gerar corretamente uma tradução se todos os construtores utilizados por este tradutor já tiverem sido gerados.

❖ *Manutenção das Assertivas de Dependência Existencial*

O passo 2 do algoritmo *R1* define as atualizações requeridas para a manutenção das assertivas de dependência existencial. Como as assertivas de dependência existencial são as mesmas para tipos de entidade e tipos de relacionamento, temos que a ação requerida para a manutenção destas assertivas na adição em um tipo de relacionamento são estas apresentadas na tabela 3 do Apêndice B.

❖ *Manutenção das Assertivas de Correspondência de Caminhos*

O passo 3 do algoritmo *R1* define as atualizações requeridas para a manutenção das assertivas de correspondência de caminhos. Uma assertiva de correspondência de caminhos da forma $\boldsymbol{R_M}$-$\boldsymbol{A_M} \equiv \boldsymbol{Y_1}$-$\boldsymbol{Y_2}$-$\boldsymbol{Y_3}$…-$\boldsymbol{Y_n}$-$\boldsymbol{A}$, onde $\boldsymbol{Y_1},\ldots,\boldsymbol{Y_n}$ são tipos, especifica o caminho de derivação do atributo $\boldsymbol{A_M}$. O procedimento para definir as atualizações requeridas para a manutenção das assertivas de correspondência de caminhos, na adição em um tipo de relacionamento, é o mesmo adotado para a modificação de atributos, o qual já foi explicado no capítulo 6. É importante lembrar que nos casos em que $\boldsymbol{Y_1}$ for um tipo de relacionamento ele não poderá ser o tipo escolhido para a atualização. Além disso, as assertivas de correspondência de caminhos da forma $\boldsymbol{R_M}$-$\boldsymbol{A_M} \equiv \boldsymbol{Y_1}$-$\boldsymbol{Y_2}$-$\boldsymbol{A}$, onde $\boldsymbol{Y_1}$ é um tipo de relacionamento e $\boldsymbol{A}$ é um atributo de $\boldsymbol{Y_2}$, não precisam ser consideradas neste passo, uma vez que elas são usadas no passo 1 para definir as ligações de entidade de $\boldsymbol{Y_1}$.

8.1.1 Construtores para Tipos de Relacionamento de Mediador

Como visto no capítulo 7, para adicionar uma instância em um tipo base é preciso primeiro criar a instância a ser adicionada. Estas instâncias podem ser criadas através de construtores que recebem como entrada um relacionamento de mediador e retornam uma instância semanticamente equivalente a este relacionamento de mediador.

Como um tipo de relacionamento de mediador pode ter como tipos base tanto tipos de entidade como tipos de relacionamento, precisaremos de dois tipos de construtores para criar as instâncias dos tipos base, os quais são:

- *Construtor de entidades*: recebe um relacionamento $\boldsymbol{r_M}$ de um tipo de relacionamento de mediador $\boldsymbol{R_M}$ e retorna uma entidade $\boldsymbol{e}$ de um tipo de entidade $\boldsymbol{E}$, tal que $\boldsymbol{e} \equiv \boldsymbol{r_M}$.

– *Construtor de relacionamentos*: recebe um relacionamento r_M de um tipo de relacionamento de mediador R_M e retorna um relacionamento r de um tipo de relacionamento R, tal que $r \equiv r_M$.

Para gerar estes construtores desenvolvemos dois algoritmos: o algoritmo *C3*, para gerar os construtores de entidades e o algoritmo *C4* para gerar os construtores de relacionamentos.

O algoritmo *C3*, apresentado na seção A.4, gera o construtor para criar uma entidade e de um tipo de entidade E a partir de uma especificação de relacionamento r_M = <R_M{<$E_1 : e_1$> <$E_2 : e_2$>…<$E_n : e_n$> <$A_1 : v_1$> <$A_2 : v_2$>… <$A_k : v_k$>}>, onde R_M é um tipo de relacionamento de mediador. Para criar uma entidade e a partir de r_M deve-se determinar valores para os atributos de e a partir de r_M. Como mostrado no algoritmo *C3*, os valores dos atributos de e são determinados a partir das assertivas de correspondência de atributos e de caminhos do mediador.

O algoritmo *C4*, apresentado na seção A.4, gera um construtor para criar um relacionamento r de um tipo de relacionamento R a partir de uma especificação de relacionamento r_M =<R_M{<$E_1 : e_1$>…<$E_n : e_n$> <$A_1 : v_1$>… <$A_k : v_k$>}>, onde R_M é um tipo de relacionamento de mediador. Para criar um relacionamento r a partir de r_M deve-se determinar as entidades participantes de r e os valores para os atributos de r a partir de r_M. Como mostrado no algoritmo *C4*, as entidades participantes de r e os valores para os atributos de r são determinados a partir das assertivas de correspondência de atributos e de caminhos do mediador.

8.1.2 Exemplos de Tradutores para Operações de Adição em Tipos de Relacionamento

A seguir, apresentamos um exemplo de tradutor para uma operação de adição em um relacionamento, que foi gerado usando o algoritmo *R1*.

Considere a base de dados com esquema S_1 apresentado na *Figura 8.1*.

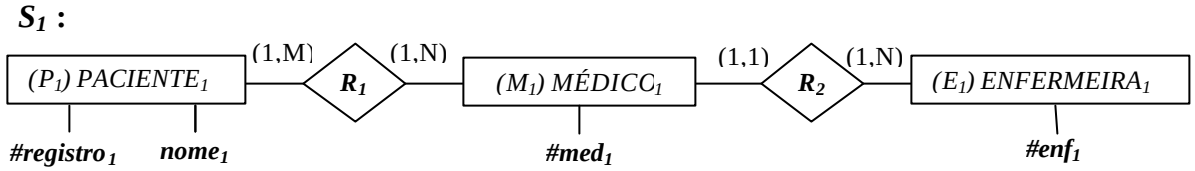


Figura 8.1 - Esquema S_1

Suponha o mediador chamado ***med₂,*** que é obtido a partir do esquema S_1. O esquema de ***med₂*** é mostrado na *Figura 8.2*.

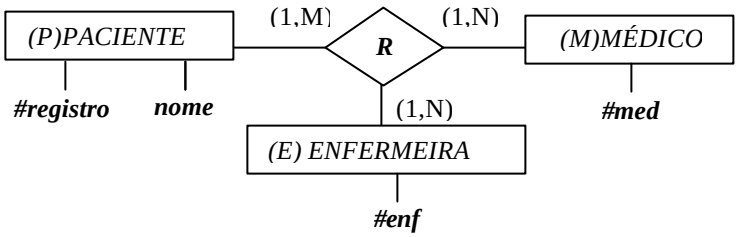


*Figura 8.2 - Esquema de **med₂***

Integrando o esquema do mediador com o esquema do banco de dados local S_1, obtemos as seguintes assertivas de correspondência:

Assertivas de Correspondência de Tipos:

AC_1: $R \equiv R_1$ AC_3: $E \equiv E_1$

AC_2: $M \equiv M_1$ AC_4: $P \equiv P_1$

Assertivas de Correspondência de Atributos:

AC_5: $\#med \equiv \#med_1$ AC_7: $\#registro \equiv \#registro_1$

AC_6: $\#enf \equiv \#enf_1$ AC_8: $nome \equiv nome_1$

Assertivas de Correspondência de Caminhos:

AC_9: $R - M \equiv R_1 - M_1$ AC_{10}: $R - P \circ R_1 - P_1$

Assertivas de Dependência Existencial:

AC_{11}: $R\ [R\text{-}E\ ,\ R - M\ ] \subseteq R_2\ [R_2 - E_1, R_2 - M_1\ ]$

O tradutor para a operação de adição em R, denominado ***Adiciona_R***, é mostrado na *Figura 8.3*. As assertivas de correspondência relevantes para esta operação, e que portanto foram consideradas pelo algoritmo *R1* foram as assertiva AC_1 e AC_{11}. As ações requeridas para estas assertivas são descritas a seguir:

- AC_1 ($R \equiv R_1$) : ação requerida para AC_1 é criar uma instância r_1 de R_1, tal que $r_1 \equiv r_M$, e em seguida, adicionar r_1 em R_1. Para criar r_1 usamos o construtor *cria_R_1_de_R*, apresentado na *Figura 8.4*. Este construtor é gerado pelo algoritmo *C4*, onde as entidades participantes de r_1 são determinadas pelas assertivas AC_9 e AC_{10}.

- AC_{11} ($R\ [R\text{-}E\ ,\ R\text{-}M\ ] \subseteq R_2\ [R_2\text{-}E_1, R_2\text{-}M_1\ ]$): como $id(R_2) \subseteq \{R_2\text{-}E_1, R_2\text{-}M_1\}$ então $R \rightarrow R_2$. A ação requerida para preservar esta restrição referencial consiste em verificar se existe algum r_2 em R_2 tal que $r_2[R_2 - E_1, R_2 - M_1] = r_M[R\text{-}E\ ,\ R\text{-}M]$. Se não existir nenhuma instância em R_2 que satisfaça essa condição então não é possível prosseguir com a atualização. Nesse caso, deve ser feito um *Roolback*.

Adiciona_ R_1 (r_M)
{
/ Ação requerida para a assertiva de correspondência AC_1 */*
r_1 : = *cria* $\_R_1\_de\_R$ (r_M);
Adicione r_1 *em* R_1;

/ Ação requerida para a assertiva de correspondência AC_{11} */*
Se não existir r_2 em R_2 tal que $r_2[R_2 – E_1, R_2 – M_1] = r[R\text{-}E, R\text{-}M]$ *então*
Roolback;

}

Figura 8.3 - Tradutor para a operação de adição em R_1.

Cria_R_1_de_R (<R{<**PACIENTE** : e_1> <**MÉDICO** : e_2> <**ENFERMEIRA** : e_3>}>)
{

/ r_M =<R{<**PACIENTE** : e_1> <**MÉDICO** : e_2> <**ENFERMEIRA** : e_3>}>, onde e_1, e_2 e e_3 são variáveis cujos valores serão passados como parâmetros em tempo de atualização */*

Obtenha p de P_1 tal que $p \equiv e_1$;

Obtenha m de M_1 tal que $m \equiv e_2$;

Retorne (< R_1{<P_1 : p> <M_1 : m>}>);
}

*Figura 8.4 - Construtor utilizado pelo Tradutor **Adiciona_R_1**.*

8.2 Definindo Tradutores para as Operações de Remoção em Tipos de Relacionamento

O algoritmo *R2*, apresentado na seção A.3, gera tradutores para as operações de remoção em tipos de relacionamento de mediador. Neste trabalho, um pedido de remoção de relacionamento é expresso por: *"Remova **r** de **R** "*, onde **r** é um relacionamento do tipo de relacionamento **R**. Denominaremos de ***Remove_R_M***, o tradutor para a operação de

remoção em um tipo de relacionamento $\boldsymbol{R_M}$. O único parâmetro passado para este tradutor é o relacionamento $\boldsymbol{r_M}$ que desejamos remover de $\boldsymbol{R_M}$.

As assertivas de correspondência relevantes para a operação de *remoção* e que portanto devem ser consideradas no algoritmo *R2* são:

- *Assertivas de Correspondência de Tipos*
- *Assertivas de Dependência Existencial*

Assim como no algoritmo *R1*, as assertivas de dependência existencial consideradas no algoritmo *R2* são apenas as assertivas que especificam restrições referenciais, os demais tipos não são tratados. O algoritmo *R2* é dividido em dois passos: o passo 1 define as atualizações requeridas para a manutenção das assertivas de correspondência de tipos e o passo 2 define as atualizações requeridas para a manutenção das assertivas de dependência existencial.

❖ *Manutenção das Assertivas de Correspondência de Tipos*

O passo 1 do algoritmo *R2* define as atualizações requeridas para a manutenção das assertivas de correspondência de tipos. A Tabela 2 do Apêndice B define as ações requeridas para a manutenção das assertivas de correspondência de tipo relacionadas a *remoção* em um tipo de relacionamento. Semelhante ao que ocorre com a operação de adição, a remoção de um relacionamento pode ocasionar tanto a remoção de relacionamentos quanto a remoção de entidades nos tipos base correspondentes. Quando a ação requerida, para preservar uma dada assertiva de correspondência de tipo, consiste em remover uma instância de um dos tipos base, deve existir uma instância deste tipo base que seja semanticamente equivalente ao relacionamento de mediador a ser removido. Como o tipo base e o tipo de relacionamento de mediador são semanticamente equivalentes, então

sempre existirá um mapeamento 1-1 entre as suas instâncias. Este mapeamento é estabelecido através de assertivas de correspondência de caminho, que estabelecem o mapeamento entre a chave do tipo base e a chave do tipo de relacionamento de mediador.

❖ *Manutenção das Assertivas de Dependência Existencial*

O passo 2 do algoritmo *R2* define as atualizações requeridas para a manutenção das assertivas de dependência existencial. Semelhante ao passo 2 do algoritmo *E2*, o procedimento para definir as atualizações requeridas para a manutenção das assertivas de dependência existencial, relacionadas à remoção em um tipo de relacionamento, estão definidos na tabela 4 do Apêndice B.

8.2.1 Exemplos de Tradutores para Operações de Remoção em Tipos de Relacionamento

A seguir, apresentamos um exemplo de tradutor para uma operação de *remoção* em um tipo de relacionamento, que foi gerado usando o algoritmo *R2*.

Considere o esquema $\boldsymbol{S_1}$, apresentado na *Figura 8.1*, o esquema do mediador $\boldsymbol{med_2}$ apresentado na *Figura 8.2* e as assertivas de correspondência obtidas da integração do esquema de $\boldsymbol{med_2}$ com o esquema $\boldsymbol{S_1}$, apresentadas na seção 8.1.2.

O tradutor para a operação de *remoção* em ***R,*** denominado ***Remove_R***, é mostrado na *Figura 8.5*. A única assertiva de correspondência relevante para esta operação, e que portanto foram consideradas pelo algoritmo *R2*, é a assertiva AC_1 ($\boldsymbol{R} \equiv \boldsymbol{R_1}$). A ação requerida para esta assertiva é verificar se existe uma instância $\boldsymbol{r_1}$ em $\boldsymbol{R_1}$ tal que $\boldsymbol{r_1} \equiv \boldsymbol{r_M}$, e no caso de existir remover $\boldsymbol{r_1}$ de $\boldsymbol{R_1}$.

Remove_ R **(** r_M **)**

{

/ Ação requerida para a assertiva de correspondência* AC_1 **/*

Se existir r_1 em R_1 tal que $r_1 \equiv r_M$ *então*

Remova r_1 *de* R_1 ;

}

Figura 8.5 - Tradutor para a operação de remoção em ***R****.*

CAPÍTULO 9

CONCLUSÕES

Neste trabalho abordamos o problema de atualização de múltiplas bases de dados através de mediadores. No nosso enfoque, são gerados tradutores para cada uma das operações de atualização de mediadores que forem permitidas. Os tradutores são definidos em tempo de projeto e são armazenados juntamente com a definição do mediador. Assim, uma vez definido o tradutor, o usuário especifica atualizações através do mediador e o tradutor as traduz em atualizações nos BDs locais, sem a necessidade de qualquer diálogo adicional.

A principal contribuição deste trabalho foi o desenvolvimento de algoritmos para geração de tradutores para as operações de atualização de mediadores. Nos algoritmos propostos, os tradutores são gerados a partir das assertivas de correspondência do mediador, que especificam formalmente os relacionamentos entre o esquema do mediador e os esquemas dos BDs locais. O uso de assertivas de correspondência nos permite provar formalmente que os tradutores gerados por nossos algoritmos produzem traduções corretas.

Os algoritmos propostos também se aplicam ao problema de Tradução de Atualização de Visão (TAV), que trata da questão de definir traduções corretas de atualizações de visões em um sistema de banco de dados centralizado. Um dos problemas com outros enfoques para TAV é que eles não usam nenhum formalismo para representar

os relacionamentos entre o esquema da visão e o esquema do banco de dados. Dessa forma, não é possível garantir que as traduções geradas pelos algoritmos propostos nesses enfoques sejam corretas.

Outra contribuição deste trabalho é a extensão do modelo ER com o formalismo necessário para representar as várias formas de correspondências entre esquemas. Nós propomos formas gerais de assertivas de dependência existencial que permitem especificar formalmente a correspondência entre conceitos com estruturas diferentes.

Os algoritmos propostos geram tradutores para as seguintes operações de atualização de mediadores: *Modificação de Atributos, Adição em Tipos de Entidade, Adição em Tipos de Relacionamento, Remoção em Tipos de Entidade e Remoção em Tipos de Relacionamento.* O algoritmo que gera tradutores para as operações de modificação de atributos trata de atributos herdados e derivados. O uso das assertivas de correspondência de caminhos para especificar o caminho de derivação de um atributo derivado, nos permitiu definir precisamente as condições nas quais é possível atualizar o valor de um atributo derivado. Caso seja possível fazer a atualização, podemos determinar as ações requeridas nos bancos de dados locais para produzir o efeito desejável de uma modificação ou atribuição de valor para um atributo derivado.

Os algoritmos *E1*, *R1*, *E2* e *R2* que geram tradutores para as operações de *adição* e *remoção* em tipos de mediador tratam de várias formas de visões, as quais são definidas pelos diferentes tipos de assertivas de correspondência de tipos (Equivalência, Seleção, Diferença, União e Interseção). Para cada uma dessas assertivas de correspondência de tipos, definimos formalmente a ação requerida para a sua preservação, por ocasião de uma *adição* ou *remoção* em um tipo de mediador. Além de fazerem a manutenção das assertivas de correspondência de tipos, os algoritmos também fazem a manutenção das restrições referenciais, que são os casos mais comuns de assertivas de dependência existencial.

Os algoritmos para as operações de adição em tipos de mediador usam construtores para criar as instâncias que devem ser adicionadas nos tipos base, como conseqüência da

adição em um tipo de mediador. Para gerar os construtores, nós desenvolvemos os algoritmos *C1*, *C2*, *C3* e *C4*, os quais utilizam as assertivas de correspondência de caminhos e de atributos do mediador para determinar como criar uma instância de um tipo base que seja semanticamente equivalente a uma determinada instância de mediador.

O uso das assertivas de dependência existencial permitiu especificar formalmente a equivalência semântica entre tipos com estruturas diferentes, como por exemplo a equivalência entre um tipo de entidade e um tipo de relacionamento. Isso tornou possível tratarmos de atualizações de mediadores onde o tipo base e o tipo de mediador têm formas estruturais diferentes. É importante notar que essas atualizações não são tratadas pelos algoritmos propostos em outros enfoques [Ling96].

Outra vantagem desse enfoque consiste em podermos fazer a manutenção das restrições de integridade, correspondentes às assertivas de correspondência do mediador, através do próprio mediador de maneira bastante eficiente. A idéia é que, sabendo-se como o mediador está relacionado com cada banco de dados local, pode-se inferir as restrições inter-bancos de dados com relação aos dados na visão do mediador. Assim sendo, o mediador pode manter todas as restrições globais relacionadas aos dados da visão do mediador. Dessa maneira, ao mantermos as restrições estabelecidas entre o mediador e os bancos de dados locais também mantemos as restrições globais que estão relacionadas com dados na visão do mediador. É válido lembrar que apenas as restrições capturadas pelas assertivas de correspondência do mediador têm garantia de serem mantidas.

Como trabalhos futuros pretendemos implementar os algoritmos propostos para geração de tradutores e construtores. Esta implementação será um dos módulos (*Definição do mapeador de atualizações de mediador*) da ferramenta para especificação de mediadores que será desenvolvida para dar suporte a metodologia que nós propomos em [Vidal97]. A arquitetura dessa ferramenta é mostrada na *Figura 9.1*. Também pretendemos adaptar os algoritmos propostos para os modelos relacional e orientado a objetos e desenvolver algoritmos para gerar tradutores para atualização de atributos multivalorados, ainda não tratado.

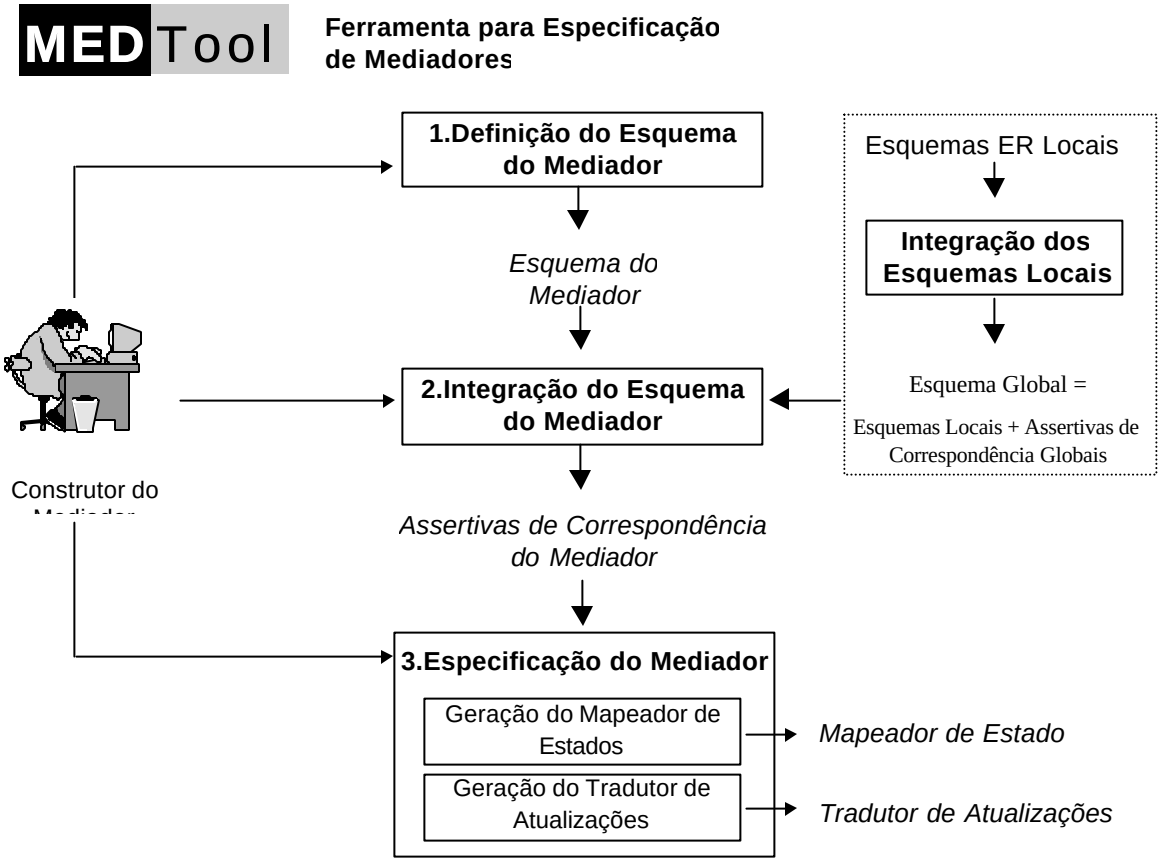


Figura 9.1 – Arquitetura da MEDTool

APÊNDICE A

A.1 Algoritmo para Atualização de Atributos

Algoritmo A1 : *Gera o tradutor para a modificação de um atributo A_M de um tipo de mediador T_M*
{
$\tau := \varnothing$;

Passo 1: *Manutenção das Assertivas de Correspondência de Atributos*

Para cada assertiva de correspondência de atributo $\boldsymbol{T_M\text{-}A_M \equiv T\text{-}A}$, onde $\boldsymbol{T}$ é um tipo local e $\boldsymbol{A}$ é um atributo de $\boldsymbol{T}$ faça:

Se $\boldsymbol{A}$ é identificador de $\boldsymbol{T}$ *então*
Retorne ($\varnothing$)*; /* Não é possível definir o tradutor. */*
senão
$\tau := \tau \cup \{<$ *Se* existir $\boldsymbol{t}$ em $\boldsymbol{T}$ tal que $\boldsymbol{t \equiv t_M}$ *então* $\boldsymbol{t.A}$:=‘$\boldsymbol{v}$’; >$\}$

Passo 2: *Manutenção das Assertivas de Correspondência de Caminho*

Para cada assertiva de correspondência de caminho Ψ da forma $\boldsymbol{T_M\text{-}A_M \equiv T_1\text{-}\ldots\text{-}T_n\text{-}A}$, onde $\boldsymbol{T_1 \ldots T_n}$ são tipos, faça:

Passo 2.1: *Determina o tipo de relacionamento $\boldsymbol{T_i}$ onde deverá ser efetuada a atualização para a manutenção de Ψ*

Se $\boldsymbol{A}$ é identificador de $\boldsymbol{T_n}$ *então*

Case $\boldsymbol{N}$ = ‘número de ligações com cardinalidade máxima >1’

$\boldsymbol{N} > 1$:
Retorne ($\varnothing$)*; /* Não é possível definir o tradutor. */*
$\boldsymbol{N} = 1$:
$\boldsymbol{T_i}$ é o tipo de relacionamento que possui a ligação com cardinalidade máxima > 1.
$\boldsymbol{N} = 0$:
Consulte o projetista para que ele determine qual o tipo de relacionamento $\boldsymbol{T_i}$ onde deverá ser efetuada a atualização.

senão

Retorne (∅); */* Não é possível definir o tradutor. */*

Passo 2.2: *Determina a atualização requerida para a manutenção de* Ψ

Case T_i

$T_i \neq T_1$ e $T_i \neq T_2$:
τ : = τ ∪{< Obtenha[2] t_1 de T_1 tal que $t_1 \equiv t_M$;
Se $t_1.(T_1\text{-}\ldots\text{-}T_n) \neq$ 'nulo' *então*
/* *Faz a modificação em* T_i */
Seja t_n a instância de T_n tal que $t_1.(T_1\text{-}\ldots\text{-}T_n) = t_n$;
Se $t_n.A \neq$ 'v' *então*
Obtenha t_i de T_i tal que $t_1.(T_1\text{-}\ldots\text{-}T_i) = t_i$;
Obtenha t_{i+1} de T_{i+1} tal que $t_{i+1}.(T_{i+1}\text{-}\ldots\text{-}T_n\text{-}A) =$ 'v';
$t_i.(T_i\text{-}\ T_{i+1}) : = \ t_{i+1}$;
senão
/* *Faz a adição em* T_i. */
Obtenha t_{i-1} de T_{i-1} tal que $t_1.(T_1\text{-}\ldots\text{-}T_{-1}) = t_{i-1}$;
Obtenha t_{i+1} de T_{i+1} tal que $t_{i+1}.(T_{i+1}\text{-}\ldots\text{-}T_n\text{-}A) =$ 'v';
Adicione $<T_i \{< T_{i-1} : t_{i-1} > <T_{i+1} : t_{i+1} >\}$ *em* T_i; > >}

$T_i = T_2$:
τ : = τ ∪{< Obtenha t_1 de T_1 tal que $t_1 \equiv t_M$;
Se $t_1.(T_1\text{-}\ldots\text{-}T_n) \neq$ 'nulo' *então*
/* *Faz a modificação em* T_2*/
Seja t_n a instância de T_n tal que $t_1.(T_1\text{-}\ldots\text{-}T_n) = t_n$;
Se $t_n.A \neq$ 'v' *então*
Obtenha t_2 de T_2 tal que $t_1.(T_1\text{-}T_2) = t_2$;
Obtenha t_3 de T_3 tal que $t_3.(T_3\text{-}\ldots\text{-}T_n\text{-}A) =$ 'v';
$t_2.(T_2\text{-}\ T_3) : = \ t_3$;
senão
/* *Faz a adição em* T_2. */
Obtenha t_3 de T_3 tal que $t_3.(T_3\text{-}\ldots\text{-}T_n\text{-}A) =$ 'v';
Adicione $<T_2 \{< T_1 : t_1 > <T_3 : t_3 >\}$ *em* T_2; > >}

$T_i = T_1$:
τ : = τ ∪{< Obtenha t_1 de T_1 tal que $t_1 \equiv t_M$;
Se $t_1.(T_1\text{-}\ldots\text{-}T_n) \neq$ 'nulo' *então*
/* *Faz a modificação em* T_1 */
Seja t_n a instância de T_n tal que $t_1.(T_1\text{-}\ldots\text{-}T_n) = t_n$;
Se $t_n.A \neq$ 'v' *então*

[2] Quando for solicitada uma instância através de um comando "obtenha" e esta instância não existir no banco de dados, então trataremos esse erro como uma *exceção*. Nesse caso, a ação corretiva é fazer um *Roolback*, visto que não é possível prosseguir com a atualização.

Obtenha t_2 de T_2 tal que $t_2.(T_2\text{-}\ldots\text{-}T_n\text{-}A) = \text{`}v\text{'}$;
$t_1.(T_1\text{-}\ T_2) :=\ t_2$;>}

Retorne (τ) ;

}*/* fim do algoritmo A1 */*

A.2 Algoritmos para Atualização de Tipos de Entidade

Algoritmo E1: *Gera o Tradutor para a Adição de uma entidade* e_M *em um Tipo de Entidade de Mediador* E_M

{

$\tau := \varnothing$;

Passo 1: *Manutenção das assertiva de correspondência de tipos*

Para cada assertiva de correspondência $E_M \equiv E$, onde E é um tipo faça:

$\tau := \tau \cup$ {< $e := cria\_E\_de\_E_M(e_M)$;

Adicione e *em* E; > }

Para cada assertiva de correspondência da forma $E \equiv E_M[P]$, onde E é um tipo e P é um predicado faça:

$\tau := \tau \cup$ {< *Se* $P(e_M)$ = 'verdade' *então*

$e := cria\_E\_de\_E_M(e_M)$;

Adicione e *em* E; > }

Para cada assertiva de correspondência da forma $E_M \equiv E_1 - E_2$, onde E_1 e E_2 são tipos faça:

$\tau := \tau \cup$ {< *Se* existir e_1 em E_1 tal que $e_1 \equiv e_M$ *então*

Se existir e_2 em E_2 tal que $e_2 \equiv e_M$ *então*

Remova e_2 *de* E_2;

senão

$e := cria\_E\_de\_E_M(e_M)$;

Se existir e_2 em E_2 tal que $e_2 \equiv e_M$ *então*

Adicione e *em* E;

Remova e_2 *de* E_2;

senão Adicione e *em* E; > }

Para cada assertiva de correspondência da forma $E_M \equiv \dot{E}_{j=1}^{k} E_j$, onde E_j é um tipo faça:

Se *SA =* E_i ***então*** ***/* O projetista será consultado para escolher o tipo*** E_i ***onde será feita a adição */***

t := t È {< e **:=** $cria\_E_i\_de\_E_M(e_M)$**;**

Adicione e *em* E_i; > }

Para cada assertiva de correspondência da forma $E_M \equiv \bigcup_{j=1}^{K} E_i$, onde E_i é um tipo faça:
t : = t È {< Para cada E_i **faça:**

e : = *cria_*E_i*_de_*E_M(e_M);
Adicione e *em* E_i; > }

Para cada assertiva de correspondência da forma $E_M \subset E$, onde E é um tipo faça:
Se ***SA*** = "Propaga" *então*
τ : = τ ∪ {< *Se* não existir e em E tal que $e \equiv e_M$ *então*
e : = *cria_*E *_de_*E_M(e_M);
Adicione e *em* E;> }
Se ***SA*** = "Bloqueia" *então*
τ : = τ ∪ { < *Se* não existir e em E tal que $e \equiv e_M$ *então*
Rollback; > }

Passo 2: *Manutenção das assertivas de dependência existencial*

Se existir uma assertiva de dependência existencial $E[C_1,\ldots,C_n] \equiv E_M[C_{M1},\ldots,C_{Mn}]$,
onde E • ® E_M *então*

Retorne (∅) */* Não é possível definir o tradutor. */*

Para cada assertiva de dependência existencial $E_M[C_{M1},\ldots,C_{Mn}] \subseteq E[C_1,\ldots,C_n]$, onde
E_M ® E faça:
τ : = τ ∪ {< *Se* não existir e em E tal que $e[C_1,\ldots,C_n] = e_M[C_{M1},\ldots,C_{Mn}]$ *então*
Roolback ; */* Não é possível fazer a atualização. */* > }

Passo 3: *Manutenção das assertivas de correspondência de caminho*

Para cada assertiva de correspondência de caminho Ψ da forma E_M-$A_M \equiv T_1$-…-T_n-A[3], onde $T_1 \ldots T_n$ são tipos, faça:

Passo 3.1: *Determina o tipo de relacionamento* T_i *onde deverá ser efetuada a atualização para a manutenção de* Ψ[4]

Se A é identificador de T_n *então*

Case N = 'número de ligações com cardinalidade máxima >1'

[3] As ACC da forma E_M-$A_M \equiv Y_1$-Y_2-A, onde Y_1 é um tipo de relacionamento e A é um atributo de Y_2, não precisam ser consideradas neste passo.
[4] Se Y_1 for um tipo de relacionamento, então ele não poderá ser escolhido para a atualização.

$N > 1$:

Retorne (∅); */* Não é possível definir o tradutor. */*

$N = 1$:

T_i é o tipo de relacionamento que possui a ligação com cardinalidade máxima > 1.

$N = 0$:

Consulte o projetista para que ele determine qual o tipo de relacionamento T_i onde deverá ser efetuada a atualização.

senão

Retorne (∅); */* Não é possível definir o tradutor. */*

Passo 3.2: *Determina a atualização requerida para a manutenção de* Ψ

Case T_i

$T_i \neq T_1$ e $T_i \neq T_2$:

$\tau := \tau \cup \{<$ Obtenha t_1 de T_1 tal que $t_1 \equiv e_M$;

Se $t_1.(T_1\text{-}\ldots\text{-}T_n) \neq$ 'nulo' *então*

/ Faz a modificação em* T_i **/*

Seja t_n a instância de T_n tal que $t_1.(T_1\text{-}\ldots\text{-}T_n) = t_n$;

Se $t_n.A \neq e_M.A_M$ *então*

Obtenha t_i de T_i tal que $t_1.(T_1\text{-}\ldots\text{-}T_i) = t_i$;

Obtenha t_{i+1} de T_{i+1} tal que $t_{i+1}.(T_{i+1}\text{-}\ldots\text{-}T_n\text{-}A) = e_M.A_M$;

$t_i.(T_i\text{-}\ T_{i+1}) := t_{i+1}$;

senão

/ Faz a adição em* T_i. **/*

Obtenha t_{i-1} de T_{i-1} tal que $t_1.(T_1\text{-}\ldots\text{-}T_{i-1}) = t_{i-1}$;

Obtenha t_{i+1} de T_{i+1} tal que $t_{i+1}.(T_{i+1}\text{-}\ldots\text{-}T_n\text{-}A) = e_M.A_M$;

Adicione $<T_i \{< T_{i-1} : t_{i-1} > <T_{i+1} : t_{i+1} > \}$ *em* $T_i; > >\}$

$T_i = T_2$:

$\tau := \tau \cup \{<$ Obtenha t_1 de T_1 tal que $t_1 \equiv e_M$;

Se $t_1.(T_1\text{-}\ldots\text{-}T_n) \neq$ 'nulo' *então*

/ Faz a modificação em* T_2 **/*

Seja t_n a instância de T_n tal que $t_1.(T_1\text{-}\ldots\text{-}T_n) = t_n$;

Se $t_n.A \neq e_M.A_M$ *então*

Obtenha t_2 de T_2 tal que $t_1.(T_1\text{-}T_2) = t_2$;

Obtenha t_3 de T_3 tal que $t_3.(T_3\text{-}\ldots\text{-}T_n\text{-}A) = e_M.A_M$;

$t_2.(T_2\text{-}\ T_3) := t_3$;

senão

/ Faz a adição em* T_2. **/*

Obtenha t_3 de T_3 tal que $t_3.(T_3\text{-}\ldots\text{-}T_n\text{-}A) = e_M.A_M$;

Adicione $<T_2 \{< T_1 : t_1 > <T_3 : t_3 > \}$ *em* $T_2; > >\}$

Retorne (τ);

}/* *fim do algoritmo E1* */

Algoritmo E2: *Gera o Tradutor para a Remoção de uma entidade* e_M *em um Tipo de Entidade de Mediador* E_M.

{

τ : = $\varnothing$;

Passo 1: *Manutenção das assertiva de correspondência de tipos*

Para cada assertiva de correspondência da forma $E_M \equiv E$, onde E é um tipo faça:

τ : = $\tau \cup$ {< *Se* existir e em E tal que $e \equiv e_M$ *então*

Remova e *de* E; >}

Para cada assertiva de correspondência da forma $E \equiv E_M[P]$, onde E é um tipo e P é um predicado faça:

Se ***SA*** = "Remove" *então*

τ : = $\tau \cup$ {< *Se* existir e em E tal que $e \equiv e_M$ *então*

Remova e *de* E; >}

Se ***SA*** = "Modifica" *então*

τ : = $\tau \cup$ {< *Se* existir e em E tal que $e \equiv e_M$ *então*

Faça $P(e)$ = 'falso'; >}

Para cada assertiva de correspondência da forma $E_M \equiv E_1 - E_2$, onde E_1 e E_2 são tipos faça:

Se ***SA*** = "Adicione" *então*

τ : = $\tau \cup$ {< *Se* existir e_1 em E_1 tal que $e_1 \equiv e_M$ *então*

Se não existir e_2 em E_2 tal que $e_2 \equiv e_M$ *então*

e_2 : = *crie_*E_2*_de_*E_M (e_M);

Adicione e_2 *em* E_2 };> }

Se ***SA*** = "Remove" *então*

τ : = $\tau \cup$ {< *Se* existir e_1 em E_1 tal que $e_1 \equiv e_M$ *então*

Se não existir e_2 em E_2 tal que $e_2 \equiv e_M$ *então*

Remova e_1 *de* E_1; > }

Para cada assertiva de correspondência da forma $E_M \equiv \bigcup_{i=1}^{k} E_i$, onde E_i é um tipo faça:
t : = t È {< Para cada E_i **faça:**

Se* existir** e_i **em** E_i **tal que** e_i **º** e_M ***então

Remova e_i ***de*** E_i**; >}**

Para cada assertiva de correspondência da forma $E_M \equiv \bigcap_{i=1}^{n} E_i$, onde E_i é um tipo faça:
Se $SA = E_i$ ***então*** */*O projetista será consultado para escolher o tipo* E_i *onde será feita a remoção*/*

t : = t È {< ***Se* existir** e_i **em** E_i **tal que** e_i **º** e_M ***então***

Remova e_i *de* E_i;>}

<u>Passo 2</u>: *Manutenção das Assertivas de Dependência Existencial*

Para cada assertiva de dependência existencial $E\ [C_{M1},\ldots,C_{Mn}] \equiv E_M\ [C_1,\ldots,C_n]$, onde $E \bullet\!\!\circledR E_M$ faça:

τ : = τ ∪ {< Para cada e em E tal que $e[C_1,\ldots,C_n] = e_M[C_{M1},\ldots,C_{Mn}]$faça:

Remova e *de* E; >}

Para cada assertiva de dependência existencial $E[C_1,\ldots,C_n] \subseteq E_M[C_{M1},\ldots,C_{Mn}]$, onde E ® E_M faça:

Se ***SA*** = "Bloqueia" *então*

τ : = τ ∪ {< *Se* existir e em E tal que $e[C_1,\ldots,C_n] = e_M[C_{M1},\ldots,C_{Mn}]$ *então*

Roolback;>}

Se ***SA*** = "Propaga" *então*

τ : = τ ∪ {< Para cada e em E tal que $e[C_1,\ldots,C_n] = e_M[C_{M1},\ldots,C_{Mn}]$ faça:

Remova e *de* E; >}

Retorne (τ) ;

}*/* fim do algoritmo E2 */*

A.3 Algoritmos para Atualização de Tipos de Relacionamento

Algoritmo R1: *Gera o Tradutor para a Adição de um relacionamento* r_M *em um Tipo de Relacionamento de Mediador* R_M

{

τ : = $\varnothing$;

Passo 1: *Manutenção das assertiva de correspondência de tipos*

Para cada assertiva de correspondência da forma $\boldsymbol{R_M \equiv E}$, onde $\boldsymbol{E}$ é um tipo faça:
τ : = $\tau \cup$ {< $\boldsymbol{e}$: = *cria_**E**_de_**R**$_M$(**r**$_M$)*;
*Adicione **e** em **E***;> }

Para cada assertiva de correspondência da forma $\boldsymbol{E \equiv R_M[P]}$, onde $\boldsymbol{E}$ é um tipo e $\boldsymbol{P}$ é um predicado faça:
τ : = $\tau \cup$ {< *Se* P (**r**$_M$) = 'verdade' *então*
$\boldsymbol{e}$: = *cria_**E**_de_**R**$_M$(**r**$_M$)*;
*Adicione **e** em **E***; > }

Para cada assertiva de correspondência $\boldsymbol{R_M \equiv E_1 - E_2}$, onde $\boldsymbol{E_1}$ e $\boldsymbol{E_2}$ são tipos faça:
τ : = $\tau \cup$ {< *Se* existir $\boldsymbol{e_1}$ em $\boldsymbol{E_1}$ tal que $\boldsymbol{e_1 \equiv e_M}$ *então*
Se existir $\boldsymbol{e_2}$ em $\boldsymbol{E_2}$ tal que $\boldsymbol{e_2 \equiv e_M}$ *então*
Remova $\boldsymbol{e_2}$ de $\boldsymbol{E_2}$;
senão
$\boldsymbol{e}$: = *cria_**E**_de_**R**$_M$(**r**$_M$)*;
Se existir $\boldsymbol{e_2}$ em $\boldsymbol{E_2}$ tal que $\boldsymbol{e_2 \equiv e_M}$ *então*
*Adicione **e** em **E***;
Remova $\boldsymbol{e_2}$ de $\boldsymbol{E_2}$};
*senão Adicione **e** em **E***; > }

Para cada assertiva de correspondência da forma $\boldsymbol{R_M \equiv \bigcup_{i=1}^{k} E_i}$, onde $\boldsymbol{E_i}$ é um tipo faça:
Se *SA* = E_i ***então*** ***/* O projetista será consultado para escolher o tipo E_i onde será feita a adição */***

t : = t È {< e **: = *cria*_E_i_*de*_R_M (**r_M**);**

*Adicione **e** em **E**$_i$*; > }

Para cada assertiva de correspondência da forma $\boldsymbol{R_M \equiv \bigcap_{i=1}^{n} E_i}$, onde $\boldsymbol{E_i}$ é um tipo faça:
t : = t È {< Para cada E_i **faça:**

$\boldsymbol{e}$: = *cria_**E** $_i$_de_**R**$_M$(**r**$_M$)*;

Adicione $\boldsymbol{e}$ *em* $\boldsymbol{E_i}$; > }

Para cada assertiva de correspondência da forma $\boldsymbol{R_M} \subset \boldsymbol{E}$, onde $\boldsymbol{E}$ é um tipo faça:

Se $\boldsymbol{SA}$ = "Propaga" *então*

τ : = $\tau \cup$ {< *Se* não existir $\boldsymbol{e}$ em $\boldsymbol{E}$ tal que $\boldsymbol{e} \equiv \boldsymbol{r_M}$ *então*

$\boldsymbol{e}$: = *cria_*$\boldsymbol{E}$ *_de_*$\boldsymbol{R_M}$($\boldsymbol{r_M}$);

Adicione $\boldsymbol{e}$ *em* $\boldsymbol{E}$; > }

Se $\boldsymbol{SA}$ = "Bloqueia" *então*

τ : = $\tau \cup$ { < *Se* não existir $\boldsymbol{e}$ em $\boldsymbol{E}$ tal que $\boldsymbol{e} \equiv \boldsymbol{r_M}$ *então*

Rollback; > }

<u>Passo 2</u>: *Manutenção das assertivas de dependência existencial*

Se existir uma assertiva de dependência existencial $\boldsymbol{E}[\boldsymbol{C_1},\ldots,\boldsymbol{C_n}] \equiv \boldsymbol{R_M}[\boldsymbol{C_{M1}},\ldots,\boldsymbol{C_{Mn}}]$,

onde $\boldsymbol{E}$ • ® $\boldsymbol{R_M}$ *então*

Retorne (∅) */* Não é possível definir o tradutor. */*

Para cada assertiva de dependência existencial $\boldsymbol{R_M}[\boldsymbol{C_{M1}},\ldots,\boldsymbol{C_{Mn}}] \subseteq \boldsymbol{E}[\boldsymbol{C_1},\ldots,\boldsymbol{C_n}]$, onde

$\boldsymbol{R_M}$ ® $\boldsymbol{E}$ faça:

τ : = $\tau \cup$ {< *Se* não existir $\boldsymbol{e}$ em $\boldsymbol{E}$ tal que $\boldsymbol{e}[\boldsymbol{C_1},\ldots,\boldsymbol{C_n}] = \boldsymbol{r_M}[\boldsymbol{C_{M1}},\ldots,\boldsymbol{C_{Mn}}]$ *então*

Roolback; */* Não é possível fazer a atualização. */* > }

<u>Passo 3</u>: *Manutenção das assertivas de correspondência de caminho*

Para cada assertiva de correspondência de caminho Ψ da forma $\boldsymbol{R_M}$-$\boldsymbol{A_M} \equiv \boldsymbol{T_1}$-…-$\boldsymbol{T_n}$-$\boldsymbol{A}$, onde $\boldsymbol{T_1}\ldots\boldsymbol{T_n}$ são tipos, faça:

<u>Passo 3.1</u>: *Determina o tipo de relacionamento* $\boldsymbol{T_i}$ *onde deverá ser efetuada a atualização para a manutenção de* Ψ

Se $\boldsymbol{A}$ é identificador de $\boldsymbol{T_n}$ *então*

Case $\boldsymbol{N}$ = 'número de ligações com cardinalidade máxima >1'

$\boldsymbol{N}$ > 1 :

Retorne (∅); */* Não é possível definir o tradutor. */*

$\boldsymbol{N}$ = 1 :

$\boldsymbol{T_i}$ é o tipo de relacionamento que possui a ligação com cardinalidade máxima > 1.

$N = 0$:

Consulte o projetista para que ele determine qual o tipo de relacionamento T_i onde deverá ser efetuada a atualização.

senão

Retorne (∅); */* Não é possível definir o tradutor. */*

Passo 3.2: *Determina a atualização requerida para a manutenção de* Ψ

Case T_i

$T_i \neq T_1$ e $T_i \neq T_2$:

$\tau := \tau \cup \{<$ Obtenha t_1 de T_1 tal que $t_1 \equiv r_M$;

Se $t_1.(T_1\text{-}\ldots\text{-}T_n) \neq$ 'nulo' *então*

/ Faz a modificação em* T_i **/*

Seja t_n a instância de T_n tal que $t_1.(T_1\text{-}\ldots\text{-}T_n) = t_n$;

Se $t_n.A \neq r_M.A_M$ *então*

Obtenha t_i de T_i tal que $t_1.(T_1\text{-}\ldots\text{-}T_i) = t_i$;

Obtenha t_{i+1} de T_{i+1} tal que $t_{i+1}.(T_{i+1}\text{-}\ldots\text{-}T_n\text{-}A) = r_M.A_M$;

$t_i.(T_i\text{-}\ T_{i+1}) := t_{i+1}$;

senão

/ Faz a adição em* T_i. **/*

Obtenha t_{i-1} de T_{i-1} tal que $t_1.(T_1\text{-}\ldots\text{-}T_{i-1}) = t_{i-1}$;

Obtenha t_{i+1} de T_{i+1} tal que $t_{i+1}.(T_{i+1}\text{-}\ldots\text{-}T_n\text{-}A) = r_M.A_M$;

Adicione $<T_i \{< T_{i-1} : t_{i-1} > <T_{i+1} : t_{i+1} > \}$ *em* $T_i; > >\}$

$T_i = T_2$:

$\tau := \tau \cup \{<$ Obtenha t_1 de T_1 tal que $t_1 \equiv r_M$;

Se $t_1.(T_1\text{-}\ldots\text{-}T_n) \neq$ 'nulo' *então*

/ Faz a modificação em* T_i **/*

Seja t_n a instância de T_n tal que $t_1.(T_1\text{-}\ldots\text{-}T_n) = t_n$;

Se $t_n.A \neq r_M.A_M$ *então*

Obtenha t_2 de T_2 tal que $t_1.(T_1\text{-}T_2) = t_2$;

Obtenha t_3 de T_3 tal que $t_3.(T_3\text{-}\ldots\text{-}T_n\text{-}A) = r_M.A_M$;

$t_2.(T_2\text{-}\ T_3) := t_3$;

senão

/ Faz a adição de um relacionamento em* T_i. **/*

Obtenha t_3 de T_3 tal que $t_3.(T_3\text{-}\ldots\text{-}T_n\text{-}A) = r_M.A_M$;

Adicione $<T_2 \{< T_1 : t_1 > <T_3 : t_3 > \}$ *em* $T_2; > >\}$

Retorne (τ);

}*/* fim do algoritmo R1 */*

Algoritmo R2: *Gera o Tradutor para a Remoção de um relacionamento* r_M *de um Tipo de Relacionamento de Mediador* R_M

{

$\tau := \emptyset$;

<u>Passo 1</u>: *Manutenção das assertiva de correspondência de tipo*

Para cada assertiva de correspondência da forma $\boldsymbol{R_M \equiv E}$, onde $\boldsymbol{E}$ é um tipo faça:

τ : = $\tau \cup$ {< *Se* existir $\boldsymbol{e}$ em $\boldsymbol{E}$ tal que $\boldsymbol{e \equiv r_M}$ *então*

Remova $\boldsymbol{e}$ *de* $\boldsymbol{E}$; >}

Para cada assertiva de correspondência da forma $\boldsymbol{E \equiv R_M[P]}$, onde $\boldsymbol{E}$ é um tipo e $\boldsymbol{P}$ é um predicado faça:

Se ***SA*** = "Remove" *então*

τ : = $\tau \cup$ {< *Se* existir $\boldsymbol{e}$ em $\boldsymbol{E}$ tal que $\boldsymbol{e \equiv r_M}$ *então*

Remova $\boldsymbol{e}$ *de* $\boldsymbol{E}$; >}

Se ***SA*** = "Modifica" *então*

τ : = $\tau \cup$ {< *Se* existir $\boldsymbol{e}$ em $\boldsymbol{E}$ tal que $\boldsymbol{e \equiv r_M}$ *então*

Faça $P(\boldsymbol{e})$ = 'falso'; >}

Para cada assertiva de correspondência da forma $\boldsymbol{R_M \equiv E_1 - E_2}$, onde $\boldsymbol{E_1}$ e $\boldsymbol{E_2}$ são tipos faça:

Se ***SA*** = "Adicione" *então*

τ : = $\tau \cup$ {< *Se* existir $\boldsymbol{e_1}$ em $\boldsymbol{E_1}$ tal que $\boldsymbol{e_1 \equiv r_M}$ *então*

Se não existir $\boldsymbol{e_2}$ em $\boldsymbol{E_2}$ tal que $\boldsymbol{e_2 \equiv r_M}$ *então*

$\boldsymbol{e_2}$: = *crie_*$\boldsymbol{E_2}$*_de_*$\boldsymbol{R_M}$ ($\boldsymbol{r_M}$);

Adicione $\boldsymbol{e_2}$ *em* $\boldsymbol{E_2}$; > }

Se ***SA*** = "Remove" *então*

τ : = $\tau \cup$ {< *Se* existir $\boldsymbol{e_1}$ em $\boldsymbol{E_1}$ tal que $\boldsymbol{e_1 \equiv r_M}$ *então*

Se não existir $\boldsymbol{e_2}$ em $\boldsymbol{E_2}$ tal que $\boldsymbol{e_2 \equiv R_M}$ *então*

Remova $\boldsymbol{e_1}$ *de* $\boldsymbol{E_1}$; > }

Para cada assertiva de correspondência da forma $\boldsymbol{R_M \equiv \dot{E}_{j=1}^{k} E_j}$, onde $\boldsymbol{E_j}$ é um tipo faça:

t : = t È {< Para cada E_i **faça:**

Se* existir** e_i **em** E_i **tal que** e_i **°** r_M ***então

Remova e_i ***de*** E_i**; >}**

$\boldsymbol{\dot{C}_{i=1}^{n}}$

Para cada assertiva de correspondência da forma $R_M \equiv E_i$, onde E_i é um tipo faça:

Se *SA* **=** E_i ***então*** ***/* O projetista será consultado para escolher o tipo*** E_i ***onde será feita a remoção */***

t : = t È {< *Se* existir e_i **em** E_i **tal que** e_i **º** r_M **e*ntão***

Remova e_i ***de*** E_i**; >}**

Passo 2: *Manutenção das Assertivas de Dependência Existencial*

Para cada assertiva de dependência existencial $R_M[C_{M1},\ldots,C_{Mn}] \equiv E[C_1,\ldots,C_n]$, onde E ® E_M faça:

τ : = τ ∪ {< Para cada ***e*** em ***E*** tal que $e[C_1,\ldots,C_n] = r_M[C_{M1},\ldots,C_{Mn}]$faça:

Remova ***e*** *de* ***E***; >}

Para cada assertiva de dependência existencial $E[C_1,\ldots,C_n] \subseteq R_M[C_{M1},\ldots,C_{Mn}]$, onde E ® E_M faça:

Se ***SA*** = "Bloqueia" *então*

τ : = τ ∪ {< *Se* existir ***e*** em ***E*** tal que $e[C_1,\ldots,C_n] = r_M[C_{M1},\ldots,C_{Mn}]$ *então*

Roolback; >}

Se ***SA*** = "Propaga" *então*

τ : = τ ∪ {< Para cada ***e*** em ***E*** tal que $e[C_1,\ldots,C_n] = r_M[C_{M1},\ldots,C_{Mn}]$ faça:

Remova ***e*** *de* ***E***; >}

Retorne (τ) ;

}*/* fim do algoritmo R2 */*

A.4 Algoritmos para Gerar Construtores

Algoritmo C1: *Gera o construtor para criar uma entidade* ***e*** *de um tipo de entidade* ***E*** *a partir de uma especificação de entidade* $<E_M \{<A_1:v_1><A_2:v_2>\ldots<A_n:v_n>\}>$, *onde* E_M *é um tipo de entidade de mediador*.

{

/ $v_1,\ldots,v_n$ *são variáveis cujos valores serão passados como parâmetro em tempo de atualização*/*

$\tau := \varnothing$;

/ Suponha* $A_1,\ldots,A_k$ *os atributos do tipo de entidade* E *.*/*

Para cada atributo A_i de E faça:

Se existir uma assertiva de correspondência da forma $E\text{-}A_i \equiv E_M\text{-}A_j$, $1 \leq j \leq n$, *então*

$\tau := \tau \cup \{< v_{A_i} := v_j; >\}$

senão

Se A_i não faz parte do identificador de E *então*

$\tau := \tau \cup \{< v_{A_i} :=$ 'nulo';>}

senão Retorne ($\varnothing$); */* O construtor não pode ser definido */*

$\tau := \tau \cup \{<$ *Retorne* $(<E\{ <A_1 : v_{A_1}> <A_2 : v_{A_2}>\ldots<A_k : v_{A_k}>\}>); >\}$

Retorne (τ) ;

}

Obs.: Os atributos derivados de E_M serão tratados no passo 3 do algoritmo *E1*, quando for feita a manutenção das assertivas de caminho.

Algoritmo C2: *Gera o construtor para criar um relacionamento* ***r*** *de um tipo de relacionamento* ***R***, *a partir de uma especificação de entidade* $<E_M \{ <A_1:v_1> <A_2:v_2> \ldots<A_n:v_n>\}>$, *onde* E_M *é um tipo de entidade de mediador*.

{

*/** $v_1,\ldots,v_n$ *são variáveis cujos valores serão passados como parâmetro em tempo de atualização */*

τ : = $\varnothing$;

/ Suponha* $E_1,\ldots,E_k$ *os tipos de entidade participando de* R *e* $A_1,\ldots,A_p$ *os atributos de* R*. */*

Passo 1: *Determina as entidades participantes de* r

Para cada tipo de entidade E_i participando de R faça :

Se existir uma assertiva de correspondência da forma $R\text{-}E_i\text{-}A_i \equiv E_M\text{-}A_j$, $1 \le j \le n$, onde $\{A_i\}$ é o identificador de E_i *então*

τ : = $\tau \cup$ {< Obtenha e_i de E_i, tal que $e_i.A_i = v_j$ >;}

senão

Se existir uma assertiva de correspondência da forma $R\text{-}E_i\text{-}\ldots\text{-}F\text{-}A \equiv E_M\text{-}A_j$, $1 \le j \le n$, onde $F \leftrightarrow E_i$ e $\{A\}$ é identificador de F *então*

τ : = $\tau \cup$ {< Obtenha f de F, tal que $f.A = v_j$;

Obtenha e_i de E_i, tal que $e_i = f.(F\text{-}\ldots\text{-}E_i)$; >}

senão

Retorne($\varnothing$); */* O construtor não pode ser definido. */*

Passo 2: *Determina os valores dos atributo de* r

Para cada atributo A_i de R faça:

Se existir uma assertiva de correspondência da forma $R\text{-}A_i \equiv E_M\text{-}A_j$, $1 \le j \le n$, *então*

τ : = $\tau \cup$ {< $v_{A_i} := v_j$; >}

senão τ : = $\tau \cup$ {< v_{A_i} := 'nulo';>}

τ : = $\tau \cup$ < *Retorne* (< R{<$E_1 : e_1$> …<$E_k : e_k$> <$A_1 : v_{A_1}$ > …<$A_p : v_{A_k}$ >}>);>

Retorne (τ) ;

}

Algoritmo C3: *Gera o construtor para criar uma entidade* ***e*** *de um tipo de entidade* ***E*** *a partir de uma especificação de relacionamento* $<R_M\{<E_1 : e_1>\ldots<E_n : e_n> <A_1 : v_1>\ldots<A_k : v_k>\}>$ *onde* R_M *é um tipo de relacionamento de mediador.*

{

*/** $e_1,\ldots,e_n$ e $v_1,\ldots,v_k$ *são variáveis cujos valores serão passados como parâmetro em tempo de atualização */*

$\tau := \varnothing$;

/ Suponha* $A_1,\ldots,A_p$ *os atributos do tipo de entidade* ***E*** *.*/*

Para cada atributo A_i de ***E*** faça:

Se existir uma assertiva de correspondência da forma $E\text{-}A_i \equiv R_M\text{-}A_j$, $1 \leq j \leq n$, *então*

$\tau := \tau \cup \{< v_{A_i} := v_j ;>\}$

senão

Se existir uma assertiva de correspondência da forma $E\text{-}A_i \equiv R_M\text{-}E_j\text{-}A$, $1 \leq j \leq n$,

então

$\tau := \tau \cup \{< v_{A_i} := e_j.A; >\}$

senão

Se existir uma assertiva de correspondência da forma $E\text{-}A_i \equiv R_M\text{-}E_j\text{-}\ldots\text{-}F\text{-}A$,

$1 \leq j \leq n$, onde $E_j \leftrightarrow F$ e $\{A\}$ é identificador de ***F*** *então*

$\tau := \tau \cup \{< v_{A_i} := e_j.(E_j\text{-}\ldots\text{-}F\text{-}A); >\}$

senão

Se A_i não faz parte do identificador de ***E*** *então*

$\tau := \tau \cup \{< v_{A_i} :=$ 'nulo';>};

senão Retorne($\varnothing$);

$\tau := \tau \cup \{ <$ *Retorne* $(< E \{ <A_1 : v_{A_1}> <A_2 : v_{A_2}>\ldots<A_p : v_{A_p}>\}>)$;

Retorne(τ);

}

Algoritmo C4: *Gera o construtor para criar um relacionamento* $\boldsymbol{r}$ *de um tipo de relacionamento* $\boldsymbol{R}$ *a partir de uma especificação de relacionamento* $<\boldsymbol{R_M}\{<\boldsymbol{E_1}:\boldsymbol{e_1}>\ldots<\boldsymbol{E_n}:\boldsymbol{e_n}><\boldsymbol{A_1}:\boldsymbol{v_1}>\ldots<\boldsymbol{A_k}:\boldsymbol{v_k}>\}>$ *onde* $\boldsymbol{R_M}$ *é um tipo de relacionamento de mediador.*

{

*/** $\boldsymbol{e_1},\ldots,\boldsymbol{e_n}$ e $\boldsymbol{v_1},\ldots,\boldsymbol{v_k}$ *são variáveis cujos valores serão passados como parâmetro em tempo de atualização */*

$\tau := \varnothing$;

/ Suponha* $\boldsymbol{E_1},\ldots,\boldsymbol{E_k}$ *os tipos de entidade participando de* $\boldsymbol{R}$ *e* $\boldsymbol{A_1},\ldots,\boldsymbol{A_p}$ *os atributos de* $\boldsymbol{R}$*. */*

<u>Passo 1</u>: *Determina as entidades participantes de* $\boldsymbol{r}$

Para cada tipo de entidade $\boldsymbol{E_i}$ participando de $\boldsymbol{R}$ faça:

Se existir uma assertiva de correspondência da forma $\boldsymbol{R}\text{-}\boldsymbol{E_i} \equiv \boldsymbol{R_M}\text{-}\boldsymbol{E_j}, 1 \le j \le n$, *então*

$\tau := \tau \cup \{<$ Obtenha $\boldsymbol{e_i}$ em $\boldsymbol{E_i}$ tal que $\boldsymbol{e_i} \equiv \boldsymbol{e_j};>\}$

senão

Se existir uma assertiva de correspondência da forma $\boldsymbol{R}\text{-}\boldsymbol{E_i}\text{-}\boldsymbol{A} \equiv \boldsymbol{R_M}\text{-}\boldsymbol{A_j}$, $1 \le j \le n$,

onde $\{\boldsymbol{A}\}$ é identificador de $\boldsymbol{E_i}$ *então*

$\tau := \tau \cup \{<$ Obtenha $\boldsymbol{e_i}$ de $\boldsymbol{E_i}$ tal que $\boldsymbol{e_i}.\boldsymbol{A} = \boldsymbol{v_j};>\}$

senão

Se existir uma assertiva de correspondência da forma $\boldsymbol{R}\text{-}\boldsymbol{E_i}\text{-}\ldots\text{-}\boldsymbol{F}\text{-}\boldsymbol{A} \equiv \boldsymbol{R_M}\text{-}\boldsymbol{A_i}$,

$1 \le j \le n$, onde $\boldsymbol{E_i} \leftrightarrow \boldsymbol{F}$ e $\{\boldsymbol{A}\}$ é identificador de $\boldsymbol{F}$ então

$\tau := \tau \cup \{<$ Obtenha $\boldsymbol{f}$ de $\boldsymbol{F}$ tal que $\boldsymbol{f}.\boldsymbol{A} = \boldsymbol{v_j}$;

Obtenha $\boldsymbol{e_i}$ de $\boldsymbol{E_i}$ tal que $\boldsymbol{e_i} = \boldsymbol{f}.(\boldsymbol{F}\text{-}\ldots\text{-}\boldsymbol{E_i});>\}$

senão

Se existir uma assertiva de correspondência da forma $\boldsymbol{R}\text{-}\boldsymbol{E_i}\text{-}\ldots\text{-}\boldsymbol{F} \equiv \boldsymbol{R_M}\text{-}\boldsymbol{E_j}$,

$1 \le j \le n$, onde $\boldsymbol{E_i} \leftrightarrow \boldsymbol{F}$ *então*

$\tau := \tau \cup \{<$ Obtenha $\boldsymbol{f}$ de $\boldsymbol{F}$ tal que $\boldsymbol{f} \equiv \boldsymbol{e_j}$;

Obtenha $\boldsymbol{e_i}$ de $\boldsymbol{E_i}$ tal que $\boldsymbol{e_i} = \boldsymbol{f}.(\boldsymbol{F}\text{-}\ldots\text{-}\boldsymbol{E_i});>\}$

senão Roolback; */*Não é possível fazer a atualização. */>*}

Passo 2: *Determina os valores dos atributo de* ***r***

Para cada atributo A_i de R faça:

Se existir uma assertiva de correspondência da forma $R\text{-}A_i \equiv R_M\text{-}A_j, 1 \leq j \leq n$, *então*

$\tau := \tau \cup \{< v_{A_i} := v_j; >\}$

senão

Se existir uma assertiva de correspondência da forma $R\text{-}A_i \equiv R_M\text{-}E_j\text{-}A$,

$1 \leq j \leq n$, onde $\{A\}$ é identificador de E_j *então*

$\tau := \tau \cup \{< v_{A_i} := e_j.A >;\}$

senão

Se existir uma assertiva de correspondência da forma $R\text{-}A_i \equiv R_M\text{-}E_j\text{-}\ldots\text{-}F\text{-}A$

$1 \leq j \leq n$, onde $E_j \leftrightarrow F$ e $\{A\}$ é identificador de F *então*

$\tau := \tau \cup \{< v_{A_i} := e_j.(E_j\text{-}\ldots\text{-}F\text{-}A) >;\}$

senão

$\tau := \tau \cup \{< v_{A_i} :=$ 'nulo';$>\}$

$\tau := \tau \cup \{<$ *Retorne* $(< R \{<E_1 : e_1>\ldots<E_k : e_k> <A_1 : v_{A_1}>\ldots<A_p : v_{A_p}>\}>);>\}$

Retorne(τ);

}

APÊNDICE B

Tabela 1 *: Preservando Assertivas de Correspondência de Tipos na adição de uma instância* e_M *em um tipo de mediador* E_M

ASSERTIVA DE CORRESPONDÊNCIA	AÇÃO REQUERIDA (AR)
E_M ° E	**AR** = <*Adicione* **e** *em* **E**>, onde $e \equiv e_M$
E_M ° $E[P]$	Se $P(e_M)$ = "True" então **AR** = <*Adicione* **e** *em* **E**>, onde $e \equiv e_M$
E_M ° $E_1 - E_2$	**Caso 1:** $\exists e_1 \in E_1$ **tal que** e_M ° $e_1 \wedge \exists e_2 \in E_2$ **tal que** e_M ° e_2 **AR** =<*Remova* e_2 *de* E_2> Caso 2: $\sim\exists e_1 \in E_1$ tal que $e_M \equiv e_1 \wedge \exists e_2 \in E_2$ tal que $e_M \equiv e_2$ ***AR* =<*Adicione* e_1 *em* E_1>, onde** e ° e_M **;<*Remova* e_2 de E_2>** Caso 3: $\sim\exists e_1 \in E_1$ tal que $e_M \equiv e_1 \wedge \sim\exists e_2 \in E_2$ tal que $e_M \equiv e_2$ **AR** =<*Adicione* e_1 *em* E_1>, onde $e_1 \equiv e_M$
E_M ° $\bigcup_{i=1}^{n} E_i$	Se **SA** = E_i então **AR** =< *Adicione* e_i *em* E_i>,onde $e_i \equiv e_M$
E_M ° $\bigcap_{i=1}^{n} E_i$	**AR** =<*Adicione* e_i *em* E_i>,onde $e_i \equiv e_M$,$\forall i \in \{1,2,\ldots,n\}$
$E_M \cup E$	Se **SA** = "Propaga" então **AR** =< Se não existir **e** em **E** tal que $e \equiv e_M$ então *Adicione* **e** *em* **E**>, onde $e \equiv e_M$ Se **SA** = "Rejeita" então **AR** =< Se não existir **e** em **E** tal que $e \equiv e_M$ então *Rollback*>

Tabela 2 : *Preservando Assertivas de Correspondência de Tipos na remoção de uma instância* e_M *de um tipo de mediador* E_M

ASSERTIVA DE CORRESPONDÊNCIA	AÇÃO REQUERIDA (AR)
E_M º E	AR = <*Remova* e *de* E>, onde $e \equiv e_M$
E_M º $E[P]$	Se SA = "Remove" então AR = <*Remova* e *de* E>, onde $e \equiv e_M$ Se SA = "Modifica" então AR = <$P(e)$: = 'falso'> , onde $e \equiv e_M$
E_M º $E_1 - E_2$	Se $\exists\ e_1 \in E_1$ tal que $e_M \equiv e_1 \wedge \sim\exists\ e_2 \in E_2$ tal que $e_m \equiv e_2$ então Se SA = "Adicione" então AR =<*Adicione* e_2 *em* E_2>, onde $e_2 \equiv e_M$ Se SA = "Remove" então AR =<*Remova* e_1 *de* E_1>, onde $e_1 \equiv e_M$
E_M º $\dot{E}_{i=1}^{n} E_i$	AR =<*Remova* e_i *de* E_i>, onde $e_i \equiv e_M$,$\forall i \in \{1,2,\ldots,n\}$
E_M º $\dot{C}_{i=1}^{n} E_i$	Se SA = E_i então AR =<*Remova* e_i *de* E_i,>, onde $e_i \equiv e_M$

Tabela 3 : *Preservando Assertivas de Dependência Existencial na adição de uma instância* e_M *em um tipo de mediador* E_M

Assertiva de Dependência Existencial	AÇÃO REQUERIDA (AR)
$E[C_1,\ldots,C_n] \equiv E_M[C_{M1},\ldots,C_{Mn}]$ $(E \bullet\!\!\circledR E_M)$	*Não é possível definir o tradutor.*
$E_M[C_{M1},\ldots,C_{Mn}] \subseteq E[C_1,\ldots,C_n]$ $(E_M \circledR E)$	**AR** = < *Se não existir* **e** *em* **E** *tal que* $\mathbf{e}[\mathbf{C_1},\ldots,\mathbf{C_n}] = \mathbf{e_M}[\mathbf{C_{M1}},\ldots,\mathbf{C_{Mn}}]$ *então* Roolback;>

Tabela 4 : *Preservando Assertivas de Dependência Existencial na remoção de uma instância* e_M *de um tipo de mediador* E_M

Assertiva de Dependência Existencial	AÇÃO REQUERIDA (AR)
$E[C_1,\ldots,C_n] \equiv E_M[C_{M1},\ldots,C_{Mn}]$ $(E \bullet\!\!\circledR E_M)$	***AR*** = < Para cada ***e*** em ***E*** tal que $e[C_1,\ldots,C_n] = e_M[C_{M1},\ldots,C_{Mn}]$ *Remova* ***e*** *de* ***E*** >
$E[C_1,\ldots,C_n] \subseteq E_M[C_{M1},\ldots,C_{Mn}]$ $(E \circledR E_M)$	*Se* ***SA*** = "Bloqueia" *então* ***AR*** = < Se existir ***e*** em ***E*** tal que $e[C_1,\ldots,C_n] = e_M[C_{M1},\ldots,C_{Mn}]$ então *Roolback*> *Se* ***SA*** = "Propaga" *então* ***AR*** = < Para cada ***e*** em ***E*** tal que $e[C_1,\ldots,C_n] = e_M[C_{M1},\ldots,C_{Mn}]$ *Remova* ***e*** *de* ***E***>

REFERÊNCIAS BIBLIOGRÁFICAS

[Bancilhon81] BANCILHON, F., SPYRATOS N. Updates Semantics and Relational View. *ACM Transactions on Database Systems*, v. 6, n. 4, dec. 1993.

[Batini84] BATINI, C., LENZERINI , M. A Methodology for Data Schema Integration in the Entity Relationship Model. *IEEE Transaction on Software Engineering (TSE).* v. 10, n. 6, p. 650-664, nov. 1984.

[Batini et al.86] BATINI, C., LENZERINI, M., NAVATHE, S.,B. A Comparative analyses of methodologies for database schema integration. *ACM Computer Surveys*. v. 18, p.323-364, 1986.

[Carlson79] CARLSON C.R., ARORA A. K. The Updatability of Relational Views based on Functional Dependencies. In *Proc. of Third Intl. Computer Software and Applications Conference.* nov. 1979.

[Chawathe et al.94] CHAWATHE, S., GARCIA MOLINA, H., HAMMER, J. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In *Proc. of 10th Meeting of the Information Processing Society of Japan (IPSJ)*. oct. 1994.

[Chen76] CHEN, P.P. The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems.* v.1, n.1, p. 166-192, 1976.

[Cosmadakis84] COSMADAKIS, S.,S., PAPADIMITRIOU, C.,H. Updates of Relational Views. *Journal of the ACM.* v.31, n.4, oct. 1984

[Dayal82] DAYAL U., BERNSTEIN A. On the Correct Translation of Update Operations on Relational Views. *ACM Transactions on Database Systems.* v.7, n.3, sep. 1982.

[Elmasri94] ELMASRI, R., NAVATHE, S.B. *Fundamentals of Database Systems*, 2. ed. Addison - Wesley, 1994.

[Furtado85] FURTADO A. L., CASANOVA M. A. Updating Relational Views. *Query Processing in Database Systems.* New York: Ed. Springer Verlag, 1985.

[Garcia-Molina et al.97] GARCIA-MOLINA, H., PAPAKONSTATINOU, Y., QUASS, D., RAJARAMAN, A., SAGIV, Y., ULLMAN, J., VASALOS, V., WIDOM, J. The TSIMMIS Approach to Mediation:Data Models and Languages. *Journal of Intelligent Information System*. 1997.

[Keller86a] KELLER, A.M. The Role of Semantics in Translating View Updates. *IEEE Computer*. v.19, n.1, p.63-73, jan. 1986.

[Keller86b] KELLER, A.M. Choosing a View Update Translator by Dialog at View Definition Time. In *Proc. of 12th Intl. Conf. on Very Large Databases.* 1986.

[Langerak90] LANGERAK, R. View Updates in Relational Databases with an Independent Schema. *ACM Trans. Databse Systems.* v. 15, n. 1, p.40-46, mar. 1990.

[Larson et al.89] LARSON, J.A., NAVATHE, S.B., ELMASRI, R., A Theory of Attribute Equivalence in Databases with Application to Schema Integration. *IEEE Transactions on Software Engineering*. v. 15, n.4, apr. 1989.

[Larson91] LARSON, J.A., SHETH, A.,P. Updating Relational Views using Knowledge at View Definition and View Update Time. *Information Systems.* v. 16, n. 2, p. 145-168, 1991.

[Ling87] LING, T.W., A Three Level Schema Architecture ER-based Database Management Systems. In *Proc. of the Sixth International Conference on Entity-Relationship Approach*. New York, p.205-220, nov. 1987.

[Ling91] LING, T.W., LEE, M.L., A Prolog Implementation of an ER-based DBMS. In *Proc. of 10th International Conference on ER Approach.* California, oct. 1991.

[Ling96] LING, T.W., LEE, M.L. View Update in Entity-Relationship Approach, *Data & Knowledge Engineering*. v. 19, p. 135-169, 1996.

[Masunaga84] MASUNAGA, Y. A Relational Database View Update Translation Mechanism. In *Proc. of the 10th International Conference on Very Large Databases*. Singapore, aug. 1984.

[Medeiros85] MEDEIROS, C.M.B., TOMPA, F. W. Understanding the Implications of View Update Policies. In *Proc. of 11th International Conference on Very Large Databases*. Stockholm, 1985.

[Navathe et al.86] NAVATHE, S. B., ELMASRI, R., LARSON, J. Integrating User Views in Database Design. *IEEE Computer*. v. 19, n. 1, p. 50-62, jan. 1986.

[Navathe96] NAVATHE, S. B., SAVASERE, A. A Schema Integration Facility using Object-Oriented Model. *Object Oriented Multidatabase Systems.* Prentice Hall, 1996. Capítulo 04.

[Papakonstantinou et al.94] PAPAKONSTANTINOU, Y., GARCIA-MOLINA, H., WINDOM, J. Object Exchange Across Heterogeneous Information Sources. In *Proc. of the Eleventh International Conference on Data Engineering (IEEE Computer Society)*. Taipei, Taiwan, p. 6-10, mar. 1995.

[Papakonstantinou et al.96] PAPAKONSTANTINOU, Y., GARCIA-MOLINA, H., ULLMAN, J. MedMaker: A Mediation System Based on Declarative Specifications. In *Proc. of International Conference on Data Engineering.* mar. 1996.

[Savasere et al.91] SAVASERE, A., SHETH, A., Gala, S., Navathe, S. and Marcus, H. On Applying Classification to Schema Integration. In *Proc. of the First International Workshop on Interoperability in Multidatabase Systems.* Kyoto, apr. 1991.

[Sheth90] SHETH, A., LARSON, J. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*. v. 22, n. 3, p.183-236, sep.1990.

[Spaccapietra et al.91] SPACCAPIETRA, S., PARENT, C., DUPONT, Y. Model Independent Assertions for Integration of Heterogeneous Schemas. *Very Large Databases Journal (VLDB)*, v. 1, n.1, p. 81-126, jul. 1992.

[Spaccapietra94] SPACCAPIETRA, S., PARENT, C. View Integration: A Step Forward in Solving Structural Conflicts. *IEEE Trans. on Software Engineering.* v. 6, n. 2, apr. 1994.

[Subrahmanian et al.95] SUBRAHMANIAN, V.S., ADALI, S., BRINK , A., EMERY, R., LU, J., RAJPUT, A., ROGERS, T.J., ROSS, R., WARD, C. HERMES: A Heterogeneous Reasoning and Mediator System. Technical Report, University of Maryland, 1995.

[Vidal90] VIDAL V.M.P., WINSLETT, M. Specifying Updatable Views in Object-Oriented Models. In *Proc. of V Simpósio Brasileiro de Banco de Dados.* Rio de Janeiro, apr. 1990.

[Vidal94] VIDAL, V.M.P. *Preservando a Semântica das Atualizações na Integração de Visões*. PhD thesis, COPPE/UFRJ, 1994

[Vidal94] VIDAL, V.M.P., WINSLETT, M. Preserving Update Semantics During Schema Integration. In *Proc. of Third International Conference on Information and Knowledege Management*. Gaithersburg, USA, nov. 1994.

[Vidal96] VIDAL, V. M. P., Definindo Traduções Corretas de Atualizações de Visões na Presença de Efeitos Colaterais. In *Proc. of XI Simpósio Brasileiro de Banco de Dados*. São Carlos, oct. 1996.

[Vidal97] VIDAL, V. M. P., LÓSCIO, B.F. Especificação de Mediadores para Acesso e Atualização de Múltiplas Bases de Dados. In *Proc. of XII Simpósio Brasileiro de Banco de Dados.* Fortaleza, oct. 1997.

[Wiederhold92] WIDERHOLD, G. Mediators in the Architecture of Future Information Systems. *IEEE Computer*. p.38-49, mar. 1992.

[Wiederhold94] WIEDERHOLD, G. Interoperation, Mediation and Ontologies. In *Proc. of International Symposium on Fifth Generation Computer Systems (FGCS94), Workshop on Heterogeneous Cooperative Knowledge-Bases*. v.W3, p. 33-48, ICOT, Tokyo, dec. 1994

[Wiederhold96] WIEDERHOLD, G., GENESERETH M. The Basis for Mediation. *IEEE Expert.* may. 1996.