Álisson Rabelo Arantes

WebView: Uma Ferramenta para Construção de Visões de Fontes de Dados da Web

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte

25 de julho de 2001

Resumo

A grande quantidade de dados disponíveis na Web trouxe para a comunidade de bancos de dados novos desafios para o desenvolvimento de técnicas e ferramentas para a manipulação desses dados. Dessa forma, o conceito de *visão* como um mecanismo para prover acesso a dados de interesse, desconsiderando sua estrutura, origem e formas de armazenamento tem sido reavaliado para o contexto da Web. Neste trabalho, é apresentado um ambiente composto de um conjunto de ferramentas de alto nível que permitem a coleta, extração, integração e atualização de dados provenientes da Web. Através desse ambiente, projetistas de bancos de dados podem construir e manter visões localmente materializadas, definindo esquemas para integração de dados de diferentes fontes de dados da Web e planos para atualização do conteúdo das visões.

Abstract

The huge amount of useful data available on the Web has brought to the database comunity new challenges to develop techniques and tools for the manipulation of such data. Thus, the notion of *view* as a mechanism for providing access to data of interest, regardless of their structure, origin, and form of storage, has been revisited for the Web context. In this work, we present an environment composed of a set of high-level tools that allow the fetching, extraction, integration, and refreshing of Web data. Using this environment, database designers can build and maintain locally materialized Web views by defining schemas for data integration from many distinct Web data sources and plans for refreshing the view contents.

Agradecimentos

A toda a minha família, principalmente aos meus pais Antônio e Ana, pelo infinito apoio que sempre me deram.
Aos professores Rodolfo Resende, pela orientação acadêmica e de quem fui monitor na disciplina Algoritmos e Estruturas de Dados I, Alberto Laender, pela orientação na dissertação e ao Altigran, pela co-orientação, mesmo que não oficial.
Aos demais professores e funcionários do DCC/UFMG.
Aos demais membros do grupo de Bancos de Dados do DCC/UFMG, pela convivência.
Aos meus professores da graduação da UFU, principalmente à professora Sandra de Amo, por ter me incentivado a fazer o Mestrado. Foi lá que tudo começou.
À minha Décima Quarta Turma da graduação, pela correspondência que sempre trocávamos e pelos churrascos que sempre organizávamos mas que nunca aconteciam.
A todos os amigos da famosa TurmaPos99, pelos momentos felizes que passamos juntos e pelas conversas pelos corredores quando o *stress* já "quase" tomava conta.
Aos meus amigos de Arcos, onde eu sempre me refugiava quando queria correr de tudo.
À CAPES, pelo apoio financeiro, sem o qual este trabalho não teria acontecido.

E é claro, a Deus, por nunca ter me deixado fraquejar mesmo diante dos momentos mais difíceis, e que não foram poucos.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Com a crescente popularização e disseminação da Web, um grande volume de dados tornou-se universalmente acessível para um número cada vez maior de usuários. Com isso, surgiu a necessidade do desenvolvimento de métodos e ferramentas que permitam a manipulação desses dados com funcionalidade e flexibilidade próximas do que é comumente encontrado em bancos de dados tradicionais. Estima-se que grande parte da Web é hoje composta por páginas automaticamente geradas a partir de bancos de dados, geralmente produzidas como resposta a consultas realizadas pelo preenchimento de formulários HTML [31]. Dessa forma, esses dados provêm de vários bancos de dados autônomos, o que leva à sua apresentação através de uma desorganizada coleção de páginas, espalhadas por diversos servidores nos mais diversos formatos.

É então desejável permitir que esses dados sejam acessíveis através da Web para vários tipos de aplicação. A importância do desenvolvimento dessa abordagem decorre do fato de que, atualmente, a principal forma de acesso a essas fontes de dados pelo usuário final é através da utilização de *browsers* para navegação. Esse paradigma não satisfaz adequadamente as necessidades de informação do usuário, pois o obriga a descobrir manualmente a informação desejada. Além disso, a forma como os dados são apresentados não é adequada para a realização de consultas utilizando técnicas tradicionais de bancos de dados.

Essas fontes de dados são, em sua maior parte, compostas por páginas HTML, chamadas em [16] de *páginas ricas em dados*. O conteúdo dessas páginas é composto por itens de dados, tais como classificados, horários de exibição de filmes, previsão meteorológica, resumos esportivos, índices financeiros e muitos outros, dentro de um contexto específico bem definido. Os dados contidos nessas páginas geralmente apresentam uma estrutura

irregular e sem nenhum esquema declarado, que pode apenas ser inferido de acordo com a sua apresentação. Por esse motivo, dados proveniente da Web são definidos na literatura relacionada como *dados semi-estruturados* [38]. Além de páginas HTML, essas fontes de dados são compostas por artigos de *newsgroups*, mensagens de correio eletrônico, bases de dados textuais, catálogos eletrônicos ou qualquer outro tipo de informação que possa ser acessada *on line*. Além de sua estrutura irregular, os dados disponíveis na Web apresentam a característica de estarem em constante atualização, o que aumenta ainda mais o esforço do usuário para a sua manipulação.

1.1 Descrição do Trabalho

O objetivo deste trabalho é o desenvolvimento de ferramentas que permitam ao usuário a construção e manutenção de repositórios contendo dados extraídos de fontes da Web, de forma que esses repositórios possam ser considerados visões [13, 15] localmente materializadas das fontes de dados que lhes deram origem. Essa abordagem tem sido chamada na literatura recente de *Webwarehousing* [6].

Assim como em bancos de dados tradicionais, as visões de dados da Web devem estar sempre consistentes com as fontes de dados correspondentes. As alterações que ocorrem no conteúdo das fontes devem ser automaticamente refletidas na visão. Para isso, as fontes de dados devem ser continuamente monitoradas para que, quando alguma alteração em seu conteúdo seja verificada, a mesma seja propagada para a visão, tornando-a consistente.

Diferente da abordagem de bancos de dados tradicionais, as fontes de dados da Web que constituem uma visão são mantidas independentemente e não se apresentam em estruturas de dados que possam ser diretamente manipuladas. Para a construção de tais visões, é necessário então que se tenha mecanismos eficientes para a coleta das páginas dessas fontes e para a extração dos dados das páginas coletadas. Os dados extraídos dessa forma podem então ser armazenados em estruturas de dados adequadas que permitam a sua manipulação.

O processo de coleta das páginas deve ser flexível o suficiente para tratar diversas situações, e ser capaz de coletar conjuntos de páginas estáticas ou dinâmicas. Com esse objetivo, é utilizada a ferramenta ASByE (*Agent Specification By Example*) [22, 23], cuja função é a geração de agentes para a coleta automática de conjuntos de páginas estáticas ou dinâmicas. Para o processo de extração dos dados das páginas coletadas, é utilizada a ferramenta DEByE (*Data Extraction By Example*) [29, 39]. A função dessa ferramenta é a geração de extratores de dados semi-estruturados de páginas HTML.

Ambas as ferramentas, ASByE e DEByE, assim como o trabalho aqui proposto, estão inseridos no contexto do ambiente DEByE, que é um arcabouço geral para a manipulação de dados provenientes da Web. A arquitetura geral do ambiente é apresentada pela Figura 1.1.

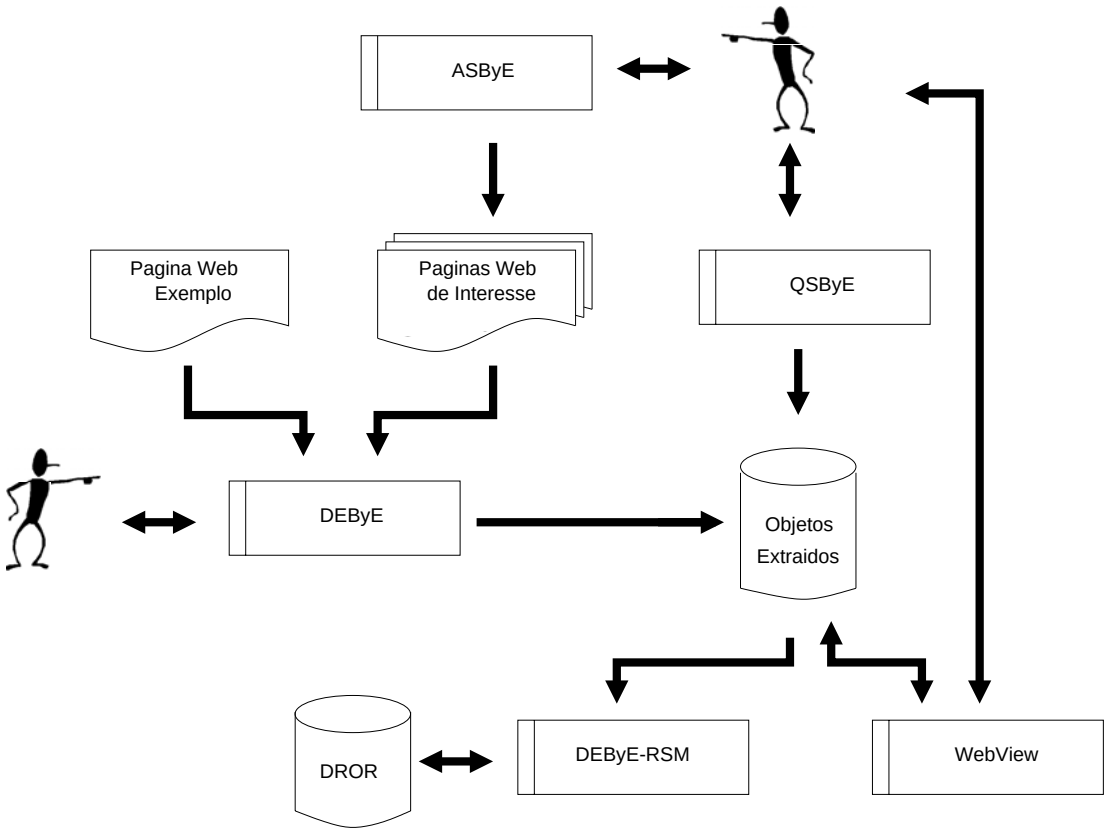


Figura 1.1: Arquitetura geral do ambiente DEByE.

Além das ferramentas ASByE e DEByE, o ambiente DEByE apresenta ainda as ferramentas QSByE (*Querying Semistructured data By Example*) [17, 18], DEByE-RSM (*DEByE - Relational Storage Manager*) [34, 35] e WebView [3, 4]. A ferramenta QSByE apresenta uma interface gráfica para consulta a dados semi-estruturados extraídos pela ferramenta DEByE, utilizando o paradigma QBE (*Query-By-Example* [45]). O módulo DEByE-RSM realiza o armazenamento dos dados semi-estruturados extraídos pela ferramenta DEByE em bancos de dados relacionais, utilizando repositórios denominados DROR (*DEByE Relational Object Repository*). Finalizando, a ferramenta WebView, objeto deste trabalho, permite a criação e manutenção de visões de fontes de dados da Web. As ferramentas ASByE, DEByE e WebView compõem o ambiente para criação e manutenção de visões proposto neste trabalho e serão apresentadas em maiores detalhes no Capítulo 3.

Uma visão de fontes de dados da Web é definida sobre um conjunto de fontes de dados de interesse $F = \{fd_1, fd_2, \ldots, fd_n\}$. A cada fonte de dados fd_i está relacionado um agente de coleta de páginas $a(fd_i)$, gerado pela ferramenta ASByE, e um extrator de dados $x(fd_i)$, gerado pela ferramenta DEByE. Os agentes e extratores são projetados especificamente de acordo com as necessidades de cada fonte de dados. Sempre que se tornar necessária a atualização da visão, os respectivos agentes de coleta são ativados, coletando as páginas das fontes de dados que alimentarão os extratores. Os dados extraídos nesse processo tornarão a visão consistente com as fontes de dados correspondentes. A Figura 1.2 ilustra de forma simplificada a arquitetura do ambiente proposto.

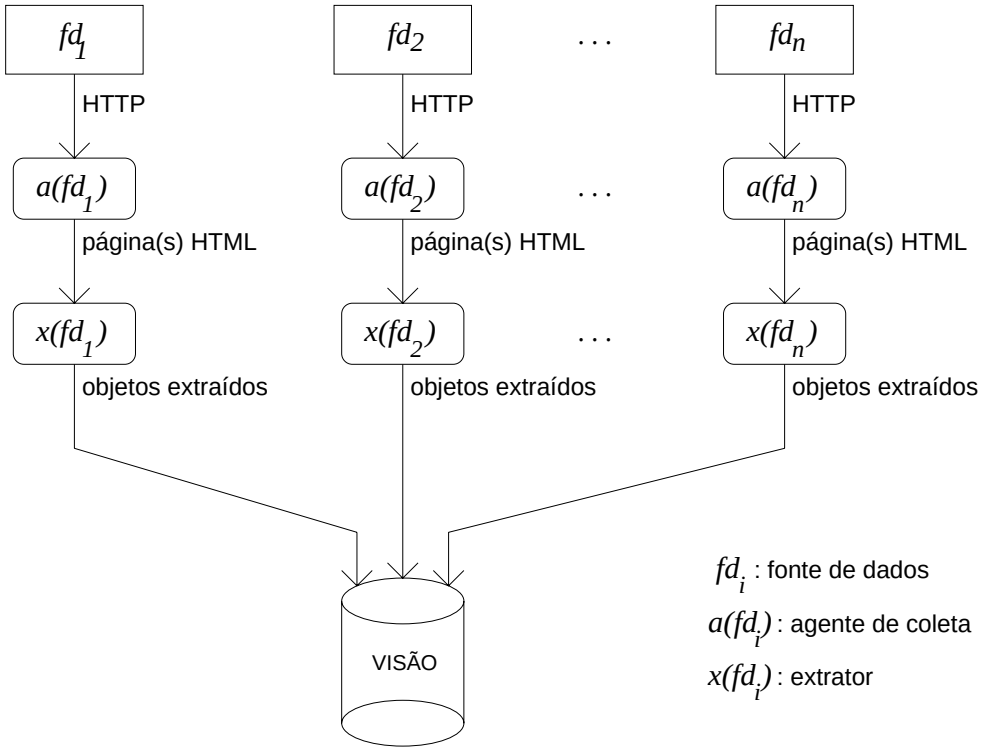


Figura 1.2: Arquitetura simplificada do ambiente proposto para materialização e manutenção de visões de fontes de dados da Web.

O ambiente desenvolvido, integrando as ferramentas ASByE e DEByE, permite ao usuário criar e manter visões localmente materializadas geradas a partir de dados provenientes da Web. Os principais desafios nessa abordagem são:

- A manutenção das visões: em virtude da natureza altamente dinâmica da Web, é necessário fazer com que as visões localmente materializadas sempre reflitam, tão fielmente quanto possível, o conteúdo atual das fontes da Web correspondentes.

- A integração de fontes de dados: para maior flexibilidade, é necessário permitir a criação de visões a partir da integração de fontes de dados distintas.

1.2 Motivação

A manutenção da consistência de visões de fontes de dados da Web se mostra extremamente necessária devido à natureza altamente dinâmica dessas fontes. Os dados extraídos da Web podem se tornar inconsistentes, desde que não representem mais o conteúdo atual das fontes de dados originais, se não houver a preocupação adequada com a sua manutenção. Além disso, as visões localmente materializadas diminuem o tempo de resposta a consultas, uma vez que eliminam a necessidade do acesso direto às fontes de dados originais, considerando que a visão reflete o conteúdo atual das fontes de dados.

Com relação ao aspecto da integração, essas visões facilitam o processo de formulação de consultas que envolvam diversas fontes de dados. Isso torna desnecessária a navegação exaustiva por várias páginas de um ou mais *sites* para que determinadas consultas sejam satisfeitas, o que decorre do fato de que elas podem integrar diversas fontes de dados distintas.

Para ilustrar a criação de uma visão de fontes de dados da Web, utilizaremos o seguinte exemplo. Sejam três fontes da Web distintas, todas relacionadas a filmes e à programação dos cinemas em Belo Horizonte. O *site* Guia de Cinema Online[1] contém uma página com todos os filmes atualmente em cartaz na cidade, como ilustra a Figura 1.3. Além disso, o *site* contém ainda outra página com informações técnicas sobre cada um dos filmes, como mostra a Figura 1.4. Já a página sobre cinemas do *site* do jornal Estado de Minas[2] contém a programação atual de cada um dos filmes em cartaz, como pode ser visto na Figura 1.5.

Considere agora que um usuário esteja interessado na seguinte informação: *"Quais os horários e salas de cinema onde está sendo exibido o filme Dr. T e as Mulheres?"*. O usuário interessado na informação completa sobre os filmes em cartaz deve navegar pelas três páginas, juntando pedaços de informação de cada uma. Por outro lado, se os dados das três páginas fossem extraídos, integrados e mantidos localmente, ao usuário bastaria a consulta a uma única página local, que conteria informação completa sobre a programação de cinema na cidade. Isso implicaria em economia de tempo, em virtude da localização dos dados, e numa maior facilidade para formulação da consulta, em virtude da integração

[1] Guia de Cinema Online - Tudo sobre os filmes em cartaz. http://www.guiadecinema.com.br
[2] Cinemas. http://cinema.uai.com.br

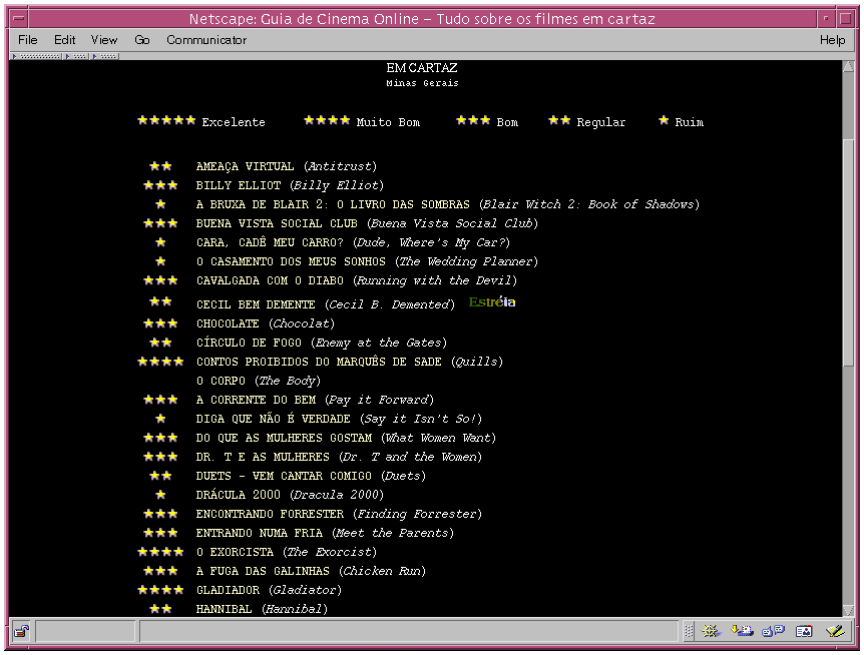


Figura 1.3: Filmes em cartaz em Belo Horizonte.

dos dados em uma única página.

Outra aplicação prática para visões de fontes de dados da Web é a sua utilização por provedores de serviços a interfaces de dispositivos móveis. Esses serviços fornecem aos dispositivos móveis dados provenientes da Web que precisam estar constantemente atualizados. Uma visão pode então ser criada utilizando as fontes de dados correspondentes para alimentar essas interfaces. Aplicações desse tipo já existem, por exemplo, para telefones celulares que permitem acesso à Web[3]. Alguns serviços disponíveis utilizam dados provenientes de várias fontes de dados, onde uma visão pode suprir todas as necessidades de informação.

Como uma última motivação, podemos imaginar um cenário onde vários computadores móveis estão interligados por uma rede sem fio. Esses computadores podem ter acesso a fontes de dados da Web localizadas em computadores de uma rede fixa. Nesse cenário de mobilidade, os principais aspectos a serem considerados são a largura de banda da conexão e o gerenciamento de energia, relacionada a baterias cuja carga é limitada [44]. Com as visões localmente materializadas nos computadores móveis, há uma diminuição do tráfego na rede móvel, economizando largura de banda e ainda o número de trocas de mensagens entre o computador móvel e a rede fixa, o que acarreta economia de energia.

[3]TodoWAP - A internet em todo lugar. http://www.todowap.com.br

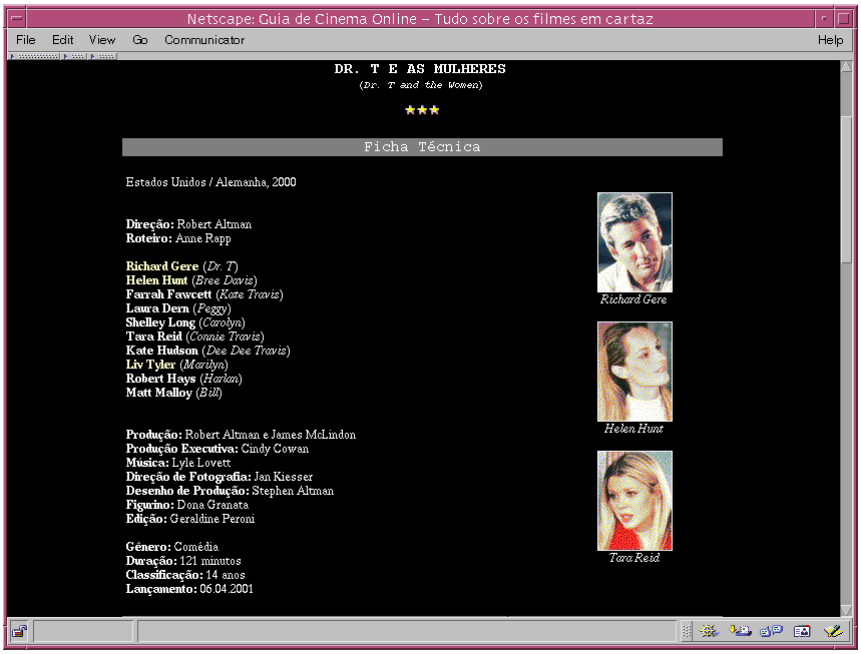


Figura 1.4: Ficha técnica dos filmes em cartaz em Belo Horizonte.

Os computadores móveis poderiam ainda estar fora de sua área de conexão, e o acesso à visão localmente materializada seria equivalente ao acesso às próprias fontes de dados da Web originais. Dessa forma, os dados estariam acessíveis mesmo quando as fontes de dados originais não estivessem.

1.3 Trabalhos Relacionados

O trabalho aqui proposto está inserido no contexto daqueles cujo objetivo principal é o desenvolvimento de métodos e ferramentas para o auxílio à modelagem, coleta, extração e integração de dados provenientes da Web, o que é atualmente chamado de *Gerência de Dados da Web* [2]. Na literatura recente, vários sistemas e ambientes têm sido propostos com esse propósito. Os principais, relacionados à abordagem proposta, são discutidos nesta seção.

O objetivo do projeto TSIMMIS [21] é o desenvolvimento de ferramentas para a integração de fontes de dados heterogêneas, tais como fontes de dados da Web. A abordagem TSIMMIS permite aos usuários a construção de mediadores que funcionam como uma camada intermediária entre as aplicações clientes e as fontes de dados heterogêneas. Os dados são coletados de diversas fontes e integrados durante a execução das consultas, ou seja, não

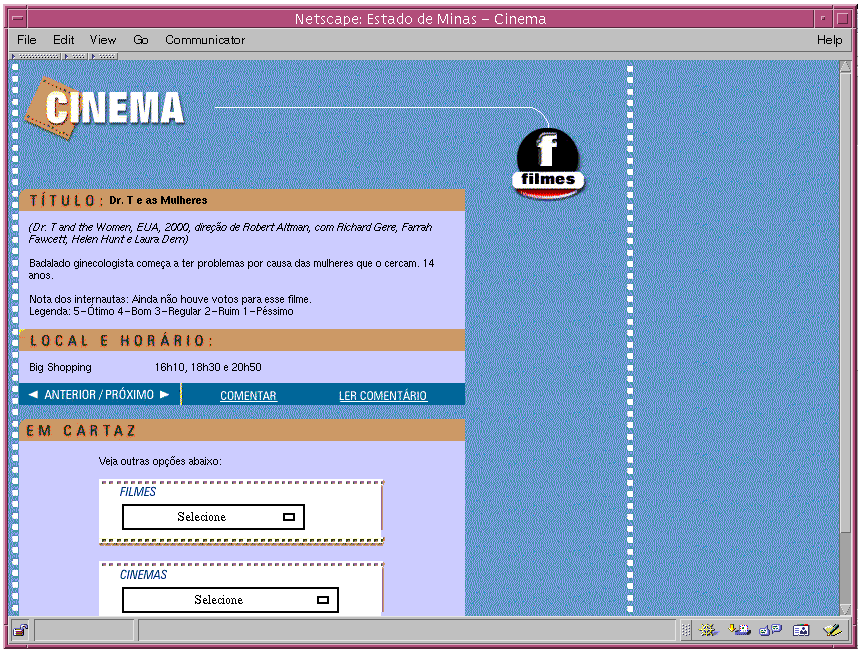


Figura 1.5: Programação dos filmes em cartaz em Belo Horizonte.

há suporte à abordagem materializada. Além disso, as ferramentas que compõem o sistema TSIMMIS requerem escrita manual de código pelo usuário para localização e extração dos dados de interesse, o que não é adequado para usuários menos especializados.

O sistema ARIADNE [27] provê ferramentas que permitem a construção de mediadores para fontes de dados da Web. Após a construção dos mediadores, esses podem então ser utilizados para a formulação de consultas aos dados disponíveis nessas fontes. Assim como nas ferramentas do ambiente TSIMMIS, os dados retornados como resposta às consultas são extraídos e integrados durante a execução da consulta. ARIADNE provê um certo grau de automação, mais próximo da abordagem aqui proposta, por apresentar ferramentas que não requerem a escrita manual de código para extração dos dados das fontes de dados da Web.

O sistema ARANEUS [36] também provê ferramentas para a construção de mediadores. Como na abordagem proposta nesse trabalho, há o suporte a visões materializadas. Entretanto, usuários devem manualmente codificar os extratores de dados das fontes da Web e as definições das visões sobre as fontes de dados. Além disso, as visões definidas no sistema ARANEUS são armazenadas em meio relacional, utilizando a linguagem ULIXES [5], que não é uma boa estratégia para o armazenamento da natureza semi-estruturada dos dados provenientes da Web.

O principal objetivo do sistema WHOWEDA (*W*are*h*ouse *o*f *We*b *Da*ta) [6] é projetar e implementar *Web warehouses* que materializam e gerenciam informações provenientes de fontes de dados da Web. No entanto, de acordo com seu modelo de dados, os dados da Web são modelados tendo as páginas e os *hyperlinks* entre elas como seus objetos primários. Dessa forma, a estrutura interna das páginas Web não é modelada, o que somente permite a formulação de consultas sobre as páginas como um todo, desconsiderando sua estrutura interna.

Em [28], Lacroix propõe um sistema para construção de visões de fontes de dados da Web. As visões são definidas através de uma linguagem de consulta, o que deve ser feito manualmente pelo usuário, e não adotam a abordagem materializada. O conteúdo de uma visão é determinado no momento em que a expressão que a define é executada. O sistema assume que para cada fonte de dados (*classe*, na sua terminologia) há uma interface CGI, cuja execução produzirá as páginas que realmente contêm os dados (*objetos*) a serem extraídos e que povoarão a visão.

O trabalho aqui proposto difere significativamente dos discutidos acima no sentido de que é totalmente baseado no uso de ferramentas de alto nível para guiar os usuários na tarefa de definição das visões. Com isso, a escrita manual de código não é necessária e, portanto, é mais conveniente para a operação por usuários menos especializados e para o rápido desenvolvimento de sistemas de informação baseados na Web. Entretanto nosso trabalho não está capacitado a manipular imagens comumente encontradas nas páginas da Web, como os sistemas ARIADNE [27] e ARANEUS [36], por exemplo.

Trabalhos existentes na área de bancos de dados federados [40] podem também ser relacionados aqui. Uma federação de bancos de dados é uma coleção de sistemas de bancos de dados preexistentes autônomos que cooperam entre si compartilhando os seus dados, sob a coordenação de um sistema gerenciados de bancos de dados federados. Dessa forma, o trabalho aqui proposto pode ser considerado como um sistema de banco de dados federados, onde os bancos de dados participantes da federação são as fontes de dados da Web. Entretanto, em bancos de dados federados, apesar da autonomia dos bancos de dados participantes da federação, existe comunicação enter eles. Com isso, alterações no conteúdo dos bancos de dados participantes são conhecidas pela federação, o que não acontece na Web.

Existem duas vertentes de trabalhos relacionados à atualização de visões em bancos de dados tradicionais. Uma delas trata da atualização do conteúdo da visão quando o banco de dados subjacente é alterado [7, 8, 9, 25]. É uma abordagem similar à abordagem aqui

proposta. Entretanto, na abordagem de bancos de dados tradicionais, o banco de dados e a visão estão sob o mesmo domínio. Dessa forma, todas as alterações ocorridas no banco de dados podem ser propagadas para as visões sobre ele definidas. Nas visões de fontes de dados da Web, as alterações sofridas pelas fontes de dados não são conhecidas pela visão. Com isso, as fontes de dados são monitoradas para a verificação de alterações em seu conteúdo, que são então propagadas para as visões. A outra vertente, por outro lado, trata da atualização do banco de dados quando o conteúdo de uma visão definida sobre ele é alterado por comandos do usuário [14, 20, 30]. As alterações ocorridas na visão devem ser propagadas para o banco de dados subjacente. Nem todas as visões são atualizáveis por comandos do usuário [13, 15] pois a tradução de uma atualização sobre a visão para uma atualização sobre o banco de dados pode ser ambígua. Esse problema não ocorre em visões de fontes de dados da Web, uma vez que o conteúdo dessas visões é apenas cópia dos dados disponíveis nas fontes.

Existe ainda outra abordagem relacionada a visões, que é a sua aplicação no contexto de dados XML [1]. XML tem se tornado um padrão para troca de dados na Web e, portanto, diferentes usuários que compartilham os mesmos dados podem ter necessidades diferentes e desejarem ver os mesmos dados sob diferentes pontos de vista. Com isso, o desenvolvimento do conceito de visões se torna também importante quando se trata de dados XML.

1.4 Contribuições da Dissertação

Neste trabalho foi desenvolvido um ambiente [3, 4] para criação e manutenção de visões de fontes de dados da Web. Através desse ambiente, o conceito de visão pode ser adotado no contexto de fontes de dados da Web. Com isso, foram desenvolvidas políticas de atualização que permitem que dados provenientes da Web sejam materializados localmente e automaticamente atualizados em relação às fontes de dados originais. Além disso, dados provenientes de fontes de dados distintas podem ser integrados em uma única visão. Essa integração facilita a formulação de consultas sobre diversas fontes de dados da Web, em virtude da integração dos dados dessas fontes em uma única visão. O resultado dessa integração é na verdade um repositório TOR (*Textual Object Repository*), um arquivo XML que implementa o modelo de dados DEByE-OM [29]. Com isso, esse resultado pode ser consultado utilizando alguma linguagem ou interface de consulta para XML [17, 32, 33], armazenado em bancos de dados relacionais [34, 35] ou republicado na Web utilizando

XSL [41].

A abordagem foi desenvolvida utilizando o conceito de visão localmente materializada. Algumas abordagens similares, entretanto, utilizam a abordagem virtual, onde os dados são coletados, extraídos e integrados durante o processamento da consulta [21, 27]. Acredita-se que a abordagem materializada é mais vantajosa, em virtude dos dados estarem materializados localmente. Com isso, as políticas de atualização dos dados provenientes da Web desenvolvidas permitem que alterações nas fontes de dados sejam refletidas nas visões. A utilização da manutenção incremental de uma visão ainda permite que se obtenha ganho em eficiência. Somente as partes alteradas das fontes de dados são propagadas para a visão, o que torna desnecessária a sua total rematerialização.

A abordagem materializada desenvolvida apresenta as seguintes vantagens: (1) diminui o tempo de resposta a consultas, em virtude da localização dos dados; (2) evita a navegação exaustiva, em virtude da integração das fontes de dados; (3) permite que os dados estejam acessíveis mesmo quando as fontes de dados originais não estejam; e (4) diminui o tráfego na rede, por eliminar a necessidade do acesso direto às fontes de dados.

O ambiente para a criação e manutenção de visões desenvolvido é totalmente baseado na utilização de ferramentas de alto nível. Dessa forma, a escrita manual de código não é necessária, o que o torna mais conveniente para a utilização por usuários menos especializados e para o rápido desenvolvimento de sistemas de informação baseados na Web, quando comparado com abordagens relacionadas.

As fontes de dados da Web, sobre as quais as visões são definidas, foram categorizadas de acordo com o seu comportamento em uma visão. Essa categorização constitui parte essencial do trabalho. Com isso, fontes de dados distintas podem ser relacionadas e os dados provenientes dessas fontes podem ser integrados. Essa categorização das fontes de dados é discutida em detalhes no Capítulo 4.

Finalmente, o trabalho desenvolvido foi realizado tendo as ferramentas ASByE e DEByE como base. Com isso, à medida que o trabalho foi se concretizando, diversas contribuições para a melhoria dessas ferramentas puderam ser detectadas. Entre essas contribuições estão o auxílio na correção de erros de implementação e melhorias nas interfaces das ferramentas. Ainda relacionado à ferramenta ASByE, diversos tipos de agentes de coleta de páginas ainda não oferecidos pela ferramenta foram detectados e adicionados à ferramenta. Além disso, houve uma maior integração das ferramentas ASByE e DEByE, que até então eram independentes.

1.5 Organização da Dissertação

O restante desta dissertação está organizado da seguinte forma. O Capítulo 2 apresenta o conceito de visão e discute a sua utilização em bancos de dados, sendo feita uma analogia entre visões em bancos de dados tradicionais e visões de fontes de dados da Web. No Capítulo 3, é descrita a arquitetura do ambiente proposto para materialização e manutenção de visões de fontes de dados da Web. Além disso, também são apresentadas em maiores detalhes as ferramentas ASByE, DEByE e WebView, que são as ferramentas do ambiente DEByE relacionadas à criação e manutenção de visões. O Capítulo 4 é dedicado a uma discussão mais detalhada das características da ferramenta WebView, que é a ferramenta desenvolvida neste trabalho. Finalmente, o Capítulo 5 faz uma revisão do trabalho desenvolvido, apresenta as principais contribuições, conclusões e possíveis direções para trabalhos futuros.

Capítulo 2

Visões

Neste capítulo, é introduzido formalmente o conceito de visão e o seu emprego na tecnologia de bancos de dados. Definido o conceito de visão, a sua utilização no contexto de bancos de dados tradicionais é discutida sendo analisadas algumas questões e problemas relacionados à sua implementação. Entre essas questões e problemas, são analisadas as estratégias adotadas para manutenção de visões, a sua utilização como mecanismo de segurança, o problema da atualização de uma visão e como uma visão pode ser implementada por um Sistema de Gerência de Bancos de Dados (SGBD). Em seguida, é discutida a utilização de visões de fontes de dados da Web, analisando-se os mesmos aspectos e fazendo-se uma analogia com as visões tradicionais.

Entende-se por bancos de dados tradicionais aqueles que são projetados e implementados especificamente para atender a determinadas aplicações, dentro do contexto específico de uma organização, utilizando-se um SGBD [13, 15]. A Web, ao contrário, embora seja hoje considerada um grande repositório de dados, não é um banco de dados. As fontes de dados da Web são implementadas e administradas independentemente cada uma voltada para uma aplicação específica. No entanto, desde que métodos e ferramentas adequados sejam desenvolvidos para a gerência de dados da Web [2], os dados da Web podem ser manipulados assim como em bancos de dados tradicionais.

Essa diferença entre bancos de dados tradicionais e fontes de dados da Web tem grande influência na discussão dos problemas citados acima. O objetivo de se fazer a analogia entre visões em bancos de dados tradicionais e visões de fontes de dados da Web, consiste em identificar na abordagem de bancos de dados tradicionais as idéias e possíveis soluções para problemas que venham a surgir no tratamento de fontes de dados da Web.

2.1 O Conceito de Visão

Na terminologia de bancos de dados relacionais[1], uma visão é uma tabela virtual originada de dados provenientes de tabelas base, que são aquelas efetivamente armazenadas no banco de dados, ou de outras visões previamente definidas [15]. Uma visão é uma tabela virtual, pois não existe por si só, e não é armazenada fisicamente no banco de dados. No entanto, isso não oferece nenhuma limitação à sua capacidade de realização de consultas. Visões podem, portanto, ser consultadas normalmente como se a consulta estivesse sendo realizada diretamente sobre as próprias tabelas que a definem.

Uma das principais vantagens de uma visão é a possibilidade de simplificar a formulação de consultas que normalmente envolvem mais de uma tabela. Considere, por exemplo, as tabelas Professor, Projeto e Trabalha_Em na Figura 2.1, do banco de dados de uma universidade fictícia.

Projeto		
Id	Nome	Localização
100	*Alfa*	*ICEx*
200	*Beta*	*ICB*
300	*Gama*	*FAFICH*

(a) Tabela Projeto

Professor		
Id	Nome	Sobrenome
20	*José*	*de Souza*
30	*Carlos*	*da Silva*
40	*Pedro*	*Cardoso*
50	*Sandra*	*Almeida*
70	*Antônio*	*Teixeira*

(b) Tabela Professor

Trabalha_Em	
Prof_Id	Proj_Id
20	*100*
20	*300*
40	*200*
30	*100*
70	*300*

(c) Tabela Trabalha_Em

Figura 2.1: Exemplos de tabelas do banco de dados de uma universidade fictícia.

Considere agora a realização de uma consulta que retorna o nome de cada professor e os nomes dos projetos nos quais ele está envolvido. Essa consulta pode ser expressa em SQL [37] da seguinte forma:

[1] O conceito de visão pode ser definido também para outras abordagens de bancos de dados [15]. Entretanto, devido à maior disseminação da abordagem relacional, nesta seção abordaremos o conceito de visão apenas no contexto do modelo de dados relacional.

```
SELECT Prof.Nome, Proj.Nome
FROM Professor Prof, Trabalha_Em TE, Projeto Proj
WHERE (Prof.Id = TE.Prof_Id) and (TE.Proj_Id = Proj.Id)
```

A formulação dessa consulta envolve duas operações de junção entre as três tabelas. Ao invés de especificar as duas operações sempre que a consulta for realizada, pode-se criar uma visão que seja definida pela própria expressão que define essa consulta. A expressão SQL abaixo expressa a definição dessa visão:

```
CREATE VIEW Prof_Proj (Professor, Projeto)
AS SELECT Prof.Nome, Proj.Nome
    FROM Professor Prof, Trabalha_Em TE, Projeto Proj
    WHERE (Prof.Id = TE.Prof_Id) and (TE.Proj_Id = Proj.Id)
```

Dessa forma, a mesma consulta pode ser realizada de forma mais simples sobre uma única tabela, que é a visão, sem a necessidade de especificar as duas operações de junção entre as três tabelas. A consulta simplificada pode ser expressa em SQL da seguinte forma:

```
SELECT Professor, Projeto
FROM Prof_Proj
```

Outra vantagem das visões é a de prover a chamada *independência lógica de dados*. A independência lógica de dados é atingida quando alterações na estrutura lógica do banco de dados se tornam transparentes para usuários e aplicações. Eventualmente, é possível que o banco de dados venha a sofrer alguma alteração em sua estrutura, de modo que, embora o conteúdo dos dados continue o mesmo, a sua alocação nas tabelas base possa mudar. Nessa situação, as consultas que eram realizadas sobre visões definidas sobre o banco de dados original permanecem inalteradas, como conseqüência da independência lógica de dados. Apenas a definição da visão precisa ser reformulada a fim de se adaptar à nova estrutura do banco de dados, desde que atributos referenciados na visão não sejam removidos.

Com a utilização de visões, a percepção do usuário sobre o banco de dados se torna simplificada. Com isso, é possível a concentração somente nos dados que são realmente de interesse, ignorando o que é irrelevante no banco de dados.

Outra grande vantagem oferecida pelas visões é a sua utilização como mecanismo de autorização e segurança. Ocasionalmente, é possível que apenas determinadas partes do banco de dados estejam acessíveis a certos grupos de usuários. Com isso, visões sobre essas partes do banco de dados podem ser criadas e apenas o direito sobre essas visões são concedidos a esses usuários. Entretanto, para visões de fontes de dados da Web, isso

não se aplica, em virtude da Web se tratar de um ambiente completamente aberto, onde o proprietário de uma visão não tem nenhum domínio sobre as fontes de dados.

Definido o conceito de visão, nas próximas seções são discutidos aspectos relacionados, analisados especificamente para visões em bancos de dados tradicionais e visões de fontes de dados da Web.

2.2 Visões em Bancos de Dados Tradicionais

Como foi discutido na seção anterior, um dos objetivos de uma visão é prover ao usuário uma visão simplificada do banco de dados, o que lhe possibilita a visualização apenas dos dados de interesse. No entanto, para que a realização de consultas sobre uma visão tenha o mesmo efeito de uma consulta sobre as tabelas que a definem, o seu conteúdo deve estar sempre consistente com o conteúdo das tabelas de definição. Todas as alterações nas tabelas sobre as quais a visão foi definida devem ser automaticamente refletidas na visão.

Com esse propósito, duas estratégias principais foram propostas para manutenção da consistência de visões em bancos de dados tradicionais: *modificação da consulta* e *materialização da visão* [15].

- **Modificação da consulta:** nessa estratégia, quando executadas consultas sobre a visão, essas são substituídas por consultas executadas diretamente sobre as tabelas que a definem [42]. A grande desvantagem dessa estratégia é a execução da consulta que define a visão sempre que uma consulta à visão for realizada. Dessa forma, as consultas podem se tornar ineficientes sobre visões definidas por consultas mais complexas, principalmente se forem realizadas dentro de um curto período de tempo. A vantagem da estratégia de modificação da consulta é que ela sempre produz resultados atualizados, uma vez que a consulta é realizada diretamente sobre as tabelas que definem a visão, o que sempre retorna o conteúdo atual do banco de dados.

- **Materialização da visão:** essa estratégia envolve a criação física de uma tabela temporária cujo conteúdo é proveniente da execução da consulta que define a visão. A estratégia de materialização da visão parte do pressuposto de que novas consultas sobre a visão serão realizadas. Com isso, a visão precisa ser eficientemente atualizada quando as tabelas de definição sofrem alterações. A visão é geralmente mantida enquanto está sendo utilizada. Se ela não for consultada por um certo período de tempo, o SGBD pode remover a tabela física temporária e rematerializar a visão quando

novas consultas surgirem. A desvantagem dessa estratégia é que a visão pode se tornar inconsistente enquanto as tabelas que a definem são atualizadas. Dessa forma, estratégias eficientes para atualização incremental da visão materializada precisam ser desenvolvidas, onde se determina quais tuplas precisam ser removidas, inseridas ou modificadas na visão materializada. A vantagem da estratégia de materialização da visão é a sua eficiência, já que as consultas são realizadas sobre uma única tabela, fisicamente armazenada.

A atualização no sentido inverso, ou seja, fazer com que atualizações na visão sejam propagadas para as tabelas que a definem é um problema ainda mais complexo [13, 14, 15, 20, 30]. Uma visão geralmente não pode ser diretamente atualizada por comandos executados pelo usuário, pois esses comandos de atualização sobre visões podem se tornar ambíguos quando mapeados para comandos que atualizem as tabelas que a definem. Como regra geral, uma visão só pode ser atualizada desde que o comando de atualização possa ser mapeado para uma única atualização nas tabelas de definição. Esse é um problema complexo e que ainda não foi totalmente resolvido, ou seja, determinar precisamente que classes de visões podem ou não ser atualizadas pelo usuário [13]. Existem duas abordagens básicas para o problema da atualização de uma visão por comandos do usuário [20]: (1) tratar a visão como um tipo abstrato de dados; e (2) tradução automática da atualização sobre a visão para uma atualização sobre as tabelas base. Na abordagem (1), a definição da visão armazena também todas as possíveis atualizações sobre ela, assim como suas traduções para atualizações sobre as tabelas base. Já a abordagem (2) deixa a cargo do SGBD a tarefa de automaticamente encontrar uma tradução da atualização que tenha o mesmo efeito se a atualização fosse executada diretamente sobre as tabelas base. O problema similar em visões de fontes de dados da Web não será discutido aqui, pois essas visões são apenas para leitura e maiores detalhes podem ser encontrados em [13, 14, 15, 20, 30].

Visões podem ainda ser utilizadas como um eficiente mecanismo para controle de autorização e segurança em bancos de dados [15]. Com a utilização desse mecanismo, os dados que não são visíveis através de determinada visão ficam protegidos contra usuários indesejados. Considere, por exemplo, as tabelas do banco de dados de uma universidade fictícia da Figura 2.1. Se a determinados usuários é permitido somente o acesso aos nomes dos professores, por exemplo, pode ser criada uma visão que inclui somente esses atributos e então somente a permissão sobre essa visão é concedida a esses usuários.

2.3 Visões de Fontes de Dados da Web

Pôde ser visto até agora a importância do conceito de uma visão, devido à sua simplicidade e, ao mesmo tempo, grande utilidade na tecnologia de bancos de dados. Devido a essa importância, o conceito de visão deve ser reavaliado no contexto da gerência de dados da Web. Com isso, podemos avaliar se as mesmas vantagens oferecidas por visões em bancos de dados tradicionais podem ser alcançadas por visões de fontes de dados da Web.

Nesta seção são discutidos os aspectos relacionados à implementação de uma visão, agora no contexto de fontes de dados da Web, como feito na seção anterior para visões em bancos de dados tradicionais. São aspectos similares à utilização de visões em bancos de dados federados [40]. Entretanto, em bancos de dados federados, embora exista a autonomia dos bancos de dados participantes, existe comunicação entre eles. Dessa forma, alterações no conteúdo dos bancos de dados participantes são percebidas pela federação, o que não acontece na Web.

De acordo com [24], existem basicamente duas abordagens para a geração de visões a partir de fontes de dados da Web: (1) a abordagem *virtual*, onde os dados são coletados, extraídos e integrados durante a execução da consulta; e (2) a abordagem *materializada*, onde os dados são extraídos, integrados e armazenados em formatos adequados para posterior manipulação consultas. As abordagens (1) e (2) podem ser associadas às estratégias de *modificação da consulta* e *materialização da visão*, respectivamente, discutidas na Seção 2.2.

A abordagem virtual não é conveniente quando o tempo de resposta às consultas é relevante para a aplicação em questão. Obter a resposta a uma consulta nessa abordagem pode ter um custo de tempo elevado, principalmente pela localização das fontes de dados. Por outro lado, na abordagem materializada, os dados materializados localmente diminuem o tempo de resposta às consultas, como já discutido na Seção 1.2. Portanto, desde que as visões sejam adequadamente mantidas em relação às fontes de dados originais, acredita-se que a abordagem materializada seja mais adequada, por fornecer melhor desempenho no tempo de execução das consultas. Por esse motivo, a abordagem materializada foi adotada neste trabalho.

Entretanto, nesse contexto, a questão da manutenção de visões é ainda mais crítica do que em bancos de dados tradicionais por dois motivos:

(a) As fontes de dados da Web podem sofrer contínuas e imprevisíveis alterações em seu conteúdo.

(b) Não existe nenhum controle sobre essas alterações, uma vez que a visão e as fontes de dados estão em domínios diferentes, ou seja, são administradas independentemente.

Em razão desses problemas, para que uma visão materializada de fontes de dados da Web possa ser mantida consistente com as fontes de dados originais, essas fontes são continuamente monitoradas. Quando alterações são verificadas no conteúdo das fontes de dados, as mesmas devem ser propagadas para a visão, num processo transparente para usuários e aplicações.

Neste trabalho, as fontes de dados são monitoradas dentro de parâmetros de freqüência especificados pelo usuário, quando a visão é criada. As políticas para atualização de visões de fontes de dados da Web utilizadas e a freqüência com que elas ocorrem serão vistas em detalhes no Capítulo 4.

Visões de fontes de dados da Web também permitem que a independência lógica de dados seja alcançada. Os agentes de coleta de páginas e os extratores de dados das páginas coletadas encapsulam as formas de acesso e armazenamento dos dados. Dessa forma, quando alterações na localização e estrutura dos dados acontece, a manutenção adequada nos agentes, extratores e no plano de materialização da visão (que será visto no Capítulo 3) permite que a estrutura final da visão permaneça inalterada. Assim, as alterações ocorridas na estrutura das fontes de dados não têm nenhuma influência nas aplicações sobre a visão, desde que os dados relativos à visão continuem na fonte de dados e que os agentes de coleta de páginas e extratores de dados sejam adequadamente mantidos.

Uma visão definida sobre fontes de dados da Web não pode ser diretamente atualizada por comandos executados pelo usuário. Os dados contidos nessas visões são apenas cópias dos dados presentes nas fontes de dados originais, com o objetivo de integrar fontes de dados distintas localmente.

Finalmente, como já foi mencionado, ao contrário de bancos de dados tradicionais, visões de fontes de dados da Web não possibilitam a sua utilização como mecanismo de autorização e segurança sobre as fontes de dados correspondentes. A Web é um universo de informação completamente aberto, onde o administrador da visão não tem nenhum controle sobre as fontes de dados correspondentes.

Finalizando a discussão, pode-se concluir que o conceito de visão, embora relativamente simples, apresenta grande importância e aplicabilidade na tecnologia de bancos de dados. Dessa forma, a sua adoção para fontes de dados da Web se mostra uma abordagem promissora. Com isso, grande parte das vantagens oferecidas pela utilização de visões em bancos de dados tradicionais podem ser alcançadas pela sua utilização sobre fontes de dados da

Web.

Capítulo 3

O Ambiente Proposto

Neste capítulo é descrita a arquitetura do ambiente proposto para a materialização e manutenção de visões de fontes de dados da Web. São descritos os módulos componentes da arquitetura e como eles se relacionam entre si. A Figura 3.1 apresenta uma visão mais detalhada dessa arquitetura. Como mencionado no Capítulo 1, o ambiente proposto é parte integrante do ambiente DEByE. A arquitetura do ambiente DEByE descrita no Capítulo 1 (Figura 1.1) forneceu uma visão mais ampla de todo o ambiente. Dos módulos componentes daquela arquitetura, os que estão relacionados a este trabalho são os módulos correspondentes às ferramentas ASByE, DEByE e WebView, que serão vistas em maiores detalhes neste capítulo.

Com o objetivo de criar uma visão de fontes de dados da Web, o usuário interage com as ferramentas ASByE [22, 23] e DEByE [29, 39] para um conjunto de fontes de dados de interesse $F = \{fd_1, fd_2, \ldots, fd_n\}$, de onde os dados que povoarão a visão serão coletados e extraídos. Para cada fonte de dados fd_i é então gerado um agente de coleta de páginas $a(fd_i)$ pela ferramenta ASByE e um extrator de dados $x(fd_i)$ pela ferramenta DEByE. Os agentes de coleta e extratores de dados são armazenados em repositórios, de onde serão invocados quando necessários para a materialização e manutenção da visão. Para cada fonte de dados fd_i é gerado um agente e um extrator, específico para essa fonte de dados. As ferramentas ASByE e DEByE, que serão descritas em maiores detalhes nas Seções 3.1 e 3.2, respectivamente, apresentam características que lhes dão a flexibilidade suficiente para a geração de agentes e extratores para uma grande variedade de fontes de dados encontradas na Web.

Após a criação dos agentes e extratores, o próximo passo é a utilização da interface

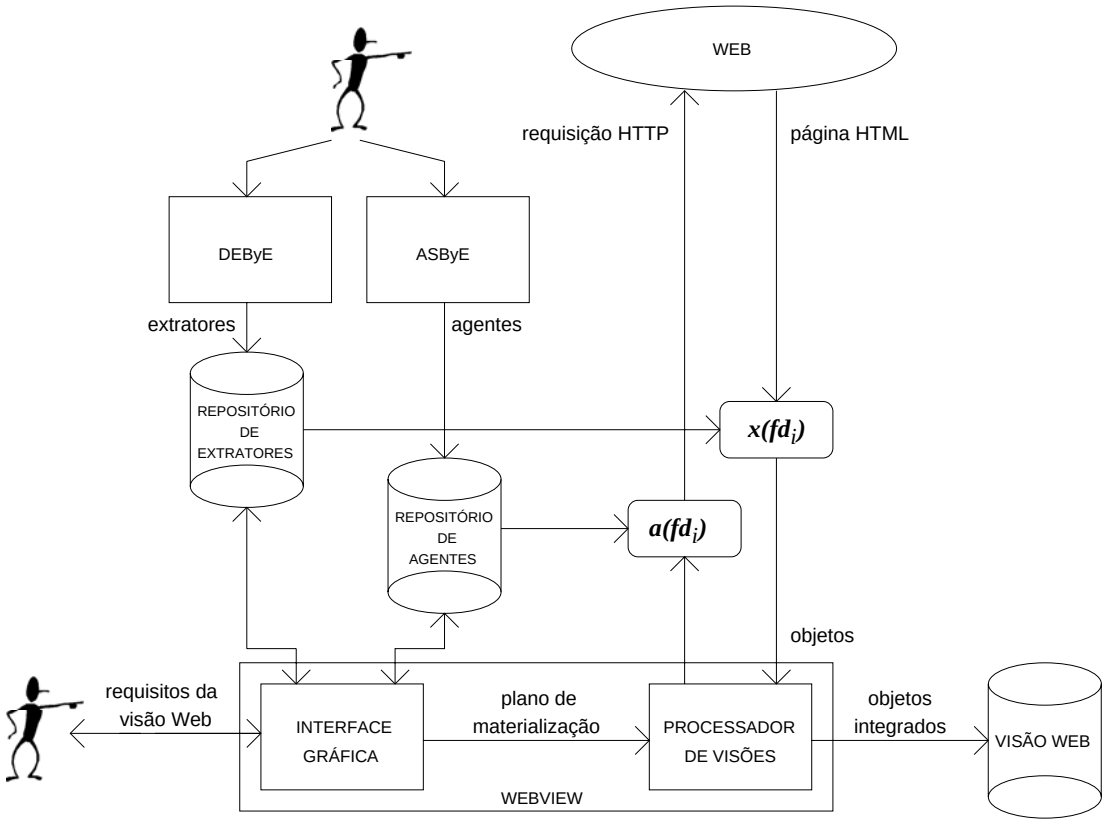


Figura 3.1: Arquitetura do ambiente proposto para materialização e manutenção de visões de fontes de dados da Web.

gráfica da ferramenta WebView. Através dessa interface gráfica, o usuário define como as fontes de dados de F estão relacionadas. Assim, é estabelecido como os dados extraídos de cada uma das fontes fd_i devem ser integrados e como eles serão atualizados em relação às fontes de dados originais.

A saída da interface gráfica da ferramenta WebView é o plano de materialização da visão. Com esse plano, o processador de visões será guiado na execução dos processos de materialização e manutenção da visão. A ferramenta WebView será descrita em maiores detalhes na Seção 3.3 e no Capítulo 4.

3.1 A Ferramenta ASByE

A ferramenta ASByE permite a geração de agentes para a coleta automática de conjuntos de páginas estáticas ou dinâmicas. A ferramenta provê uma interface de alto nível

através da qual o usuário executa uma *navegação de exemplo* em um subconjunto da Web. Com essa navegação de exemplo, o usuário informa à ferramenta quais são as fontes de dados de interesse e como alcançá-las. A Figura 3.2 ilustra a navegação de exemplo para a geração de um agente de coleta para o *site* Guia de Cinema Online[1], utilizado no exemplo da Seção 1.2.

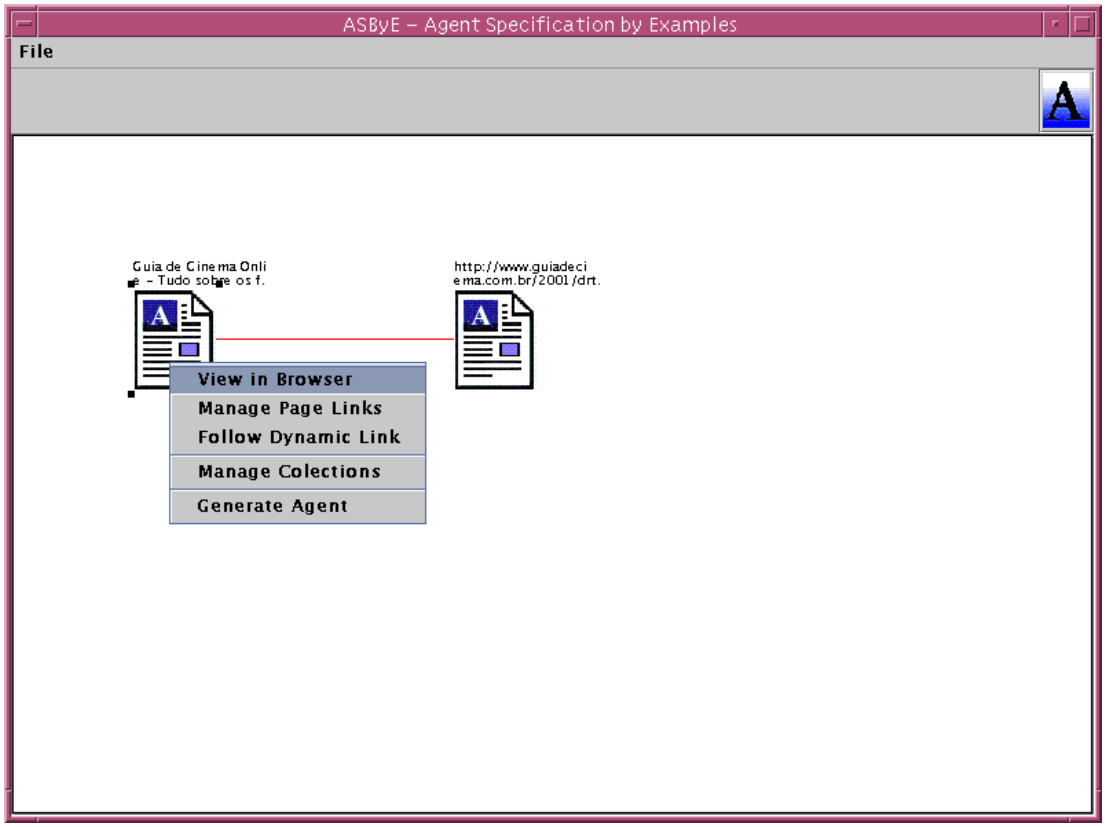


Figura 3.2: Navegação de exemplo na ferramenta ASByE para geração de um agente de coleta de páginas do *site* Guia de Cinema Online.

A interface da ferramenta ASByE apresenta uma estrutura baseada em grafos para representar o subconjunto da Web relativo às fontes de dados de interesse. Os nodos da interface representam as páginas e as arestas representam os *hyperlinks* entre elas. Existe um tipo de nodo para cada tipo de página diferente que pode ser encontrado na Web.

Uma interação típica com a ferramenta é iniciada quando o usuário fornece a URL da página de onde começará a navegação de exemplo, chamada *Web entry point* [22, 23]. A ferramenta então coleta a página correspondente e constrói o respectivo nodo na interface.

[1]Guia de Cinema Online - Tudo sobre os filmes em cartaz. http://www.guiadecinema.com.br

Dependendo do tipo de página, várias operações são oferecidas ao usuário, o que lhe permite continuar a navegação a partir deste nodo ou mesmo gerar um agente de coleta para esta página, se a página de interesse já tiver sido alcançada.

As operações permitidas para cada tipo de nodo visam fornecer uma maior flexibilidade na geração de agentes para os diversos tipos de páginas que podem ser encontrados. As principais facilidades oferecidas pela ferramenta para a coleta de páginas são descritas a seguir.

Coleções de páginas. Em algumas situações, o usuário está interessado em tratar um conjunto de páginas que estão logicamente relacionadas como uma coleção. Por exemplo, suponhamos que o usuário esteja interessado na geração de um agente para a coleta de páginas relativas aos volumes do periódico Communications of the ACM. Tais páginas podem, por exemplo, ser alcançadas a partir da página principal do *site* DB&LP[2]. A Figura 3.3 ilustra a página do *site* DB&LP que apresenta os volumes do Communications of the ACM.

A ferramenta ASByE implementa heurísticas que permitem a detecção de coleções de páginas. Com isso, quando um nodo de alguma página alcançada na navegação de exemplo é adicionado à interface da ferramenta, as heurísticas são aplicadas para verificar se existem coleções de páginas logicamente relacionadas. Se alguma coleção for detectada, a operação para geração de um agente para a coleção se torna disponível. No caso de as heurísticas falharem, o usuário pode ainda agrupar manualmente os *hyperlinks* encontrados na página em coleções. Com esta característica da ferramenta ASByE, portanto, o usuário pode proceder a geração de um único agente para a coleta de todas as páginas pertencentes à coleção.

Tratamento de formulários. Estima-se que grande parte da Web está hoje localizada na chamada *Hidden Web* [31], ou seja, páginas geradas dinamicamente por algum processo automático. Estas páginas, em sua grande maioria, são geradas pelo preenchimento de formulários HTML. Portanto, uma característica importante dos agentes de coleta é a sua capacidade de preencher automaticamente tais formulários para a coleta das páginas geradas. A ferramenta ASByE provê a geração de agentes com esta característica. Quando um nodo de uma página que contenha formulários é adicionado à interface da ferramenta, a operação *Learn to Fill Form* se torna disponível. Selecionando essa operação, a página

[2]DB&LP Bibliography. http://www.informatik.uni-trier.de/~ley/db/index.html

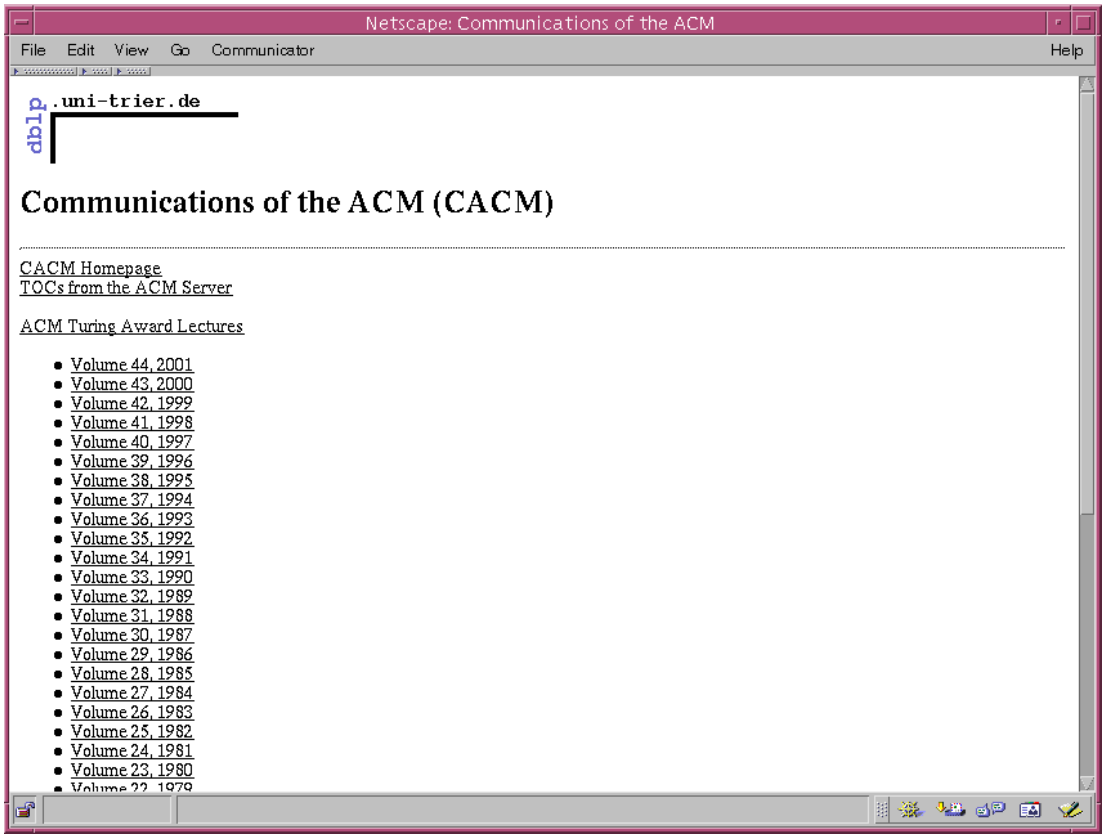


Figura 3.3: Página do periódico Communications of the ACM no *site* DB&LP.

que contém os formulários é instrumentada e mostrada através de um *browser* ao usuário. Através da página exibida pelo *browser*, todas as ações do usuário sobre a página são detectadas pela ferramenta. Dessa forma, o usuário então "ensina" à ferramenta como preencher os formulários. Os formulários podem ser preenchidos com parâmetros fixos, codificados no agente, ou ainda com parâmetros variáveis, cujos valores são fornecidos ao agente durante a sua execução. A ferramenta então codifica essa informação no agente de coleta, que preencherá corretamente os formulários necessários quando for executado. Essa característica da ferramenta ASByE é extremamente importante no tratamento de determinadas fontes de dados que podem compor uma visão de fontes de dados da Web, como será visto no Capítulo 4.

Coleta de seqüências de páginas de resposta. Em muitos casos, a resposta a uma consulta feita através de um formulário HTML é retornada como uma seqüência de páginas, cada uma das quais contendo um *hyperlink* ou um botão para a próxima página da

seqüência. Portanto, agentes para coleta das páginas geradas devem ser capazes de caminhar na seqüência até que todas as páginas retornadas tenham sido coletadas. Para nodos que representam páginas deste tipo, a ferramenta ASByE provê a operação *Learn to Follow*. Selecionando esta operação no nodo correspondente, a primeira página da seqüência é mostrada ao usuário através de um *browser*. Em um processo semelhante ao tratamento de formulários, através da página exibida pelo *browser*, o usuário também "ensina" à ferramenta como caminhar na seqüência de páginas de resposta. Assim, o agente de coleta gerado procederá de forma correta ao coletar todas as páginas da seqüência quando executado.

3.2 A Ferramenta DEByE

A função da ferramenta DEByE é a geração de extratores de dados de páginas da Web. Assim como a ferramenta ASByE, a ferramenta provê uma interface de alto nível, através da qual o usuário fornece exemplos dos objetos a serem extraídos das páginas de interesse. Os objetos de exemplo contidos nas páginas de exemplo são estruturados utilizando um paradigma baseado em tabelas aninhadas [29, 39].

A partir dos objetos de exemplo fornecidos, baseando-se no contexto sintático da página de exemplo, a ferramenta gera então um conjunto de *padrões para extração de objetos*. Os padrões para extração gerados para uma dada página de exemplo podem ser posteriormente utilizados para a extração de dados de páginas com o mesmo contexto e estrutura da página de exemplo. A Figura 3.4 ilustra uma interação com a ferramenta, onde o usuário fornece exemplos dos objetos de interesse contidos na página do *site* Guia de Cinema Online[3], utilizado no exemplo da Seção 1.2.

A interface da ferramenta é composta de duas janelas distintas: *Source Window* e *Example Window*. A janela *Source Window* nada mais é do que um *browser* embutido na ferramenta. Quando a página de exemplo é selecionada pelo usuário, esta página é exibida pela *Source Window*. A partir dessa página então, através de operações do tipo "copiar e colar", o usuário copia pedaços dos objetos contidos na página de exemplo e os cola na *Example Window*, estruturando-os em tabelas aninhadas de acordo com a sua percepção.

Após o fornecimento dos objetos de exemplo, o botão *Build Pattern* é utilizado para a geração dos padrões para extração para a página de exemplo. O botão *Extract* permite ainda executar a extração dos dados da página de exemplo utilizando os padrões previamente

[3]Guia de Cinema Online - Tudo sobre os filmes em cartaz. http://www.guiadecinema.com.br

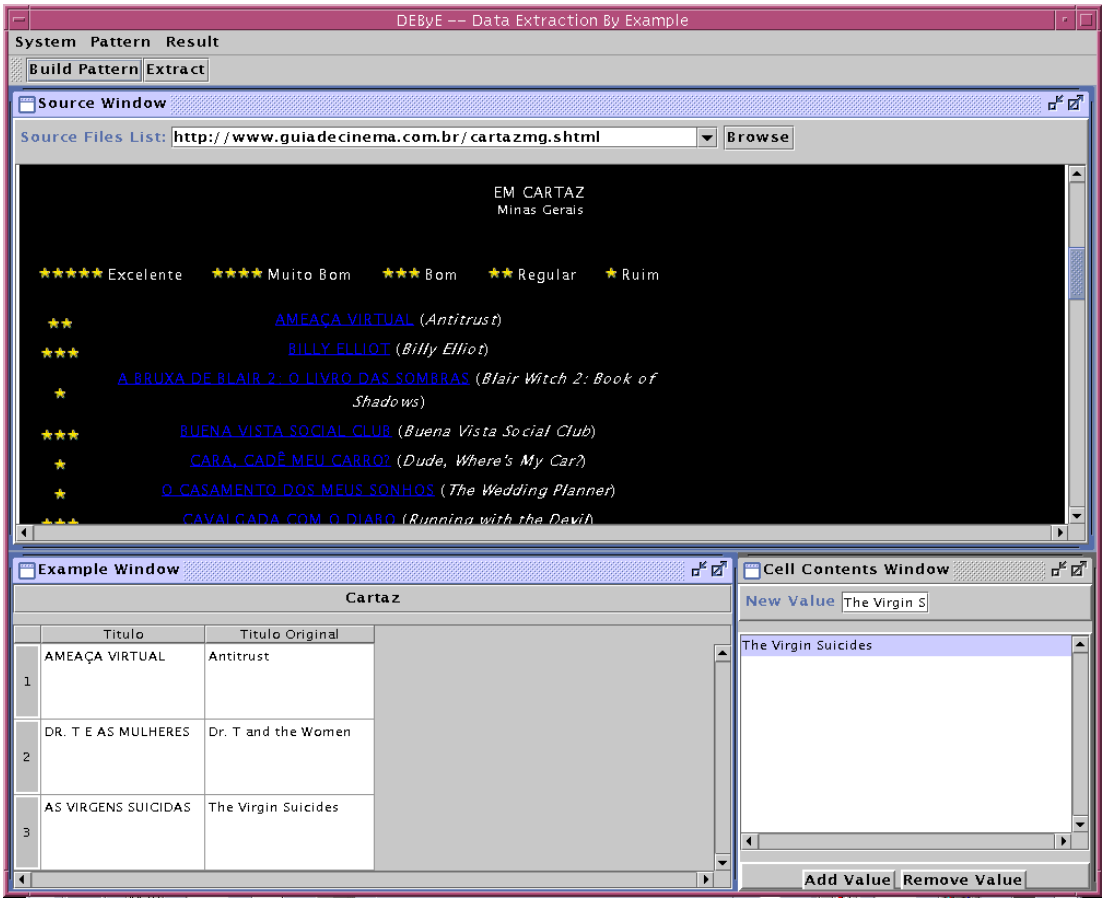


Figura 3.4: Exemplos de objetos a serem extraídos de uma página do *site* Guia de Cinema Online utilizando a ferramenta DEByE.

gerados. O resultado da extração é apresentado na janela *Extraction Result Window*. Nesse ponto, a ferramenta DEByE provê ainda uma flexibilidade denominada *Realimentação de Exemplos* [10]. Se o usuário julgar insatisfatório o resultado da extração, pode utilizar um processo interativo com a ferramenta, melhorando os exemplos fornecidos ou adicionando novos exemplos, até que o processo de extração seja considerado satisfatório. O resultado parcial do processo de extração de acordo com os exemplos fornecidos na Figura 3.4 é composto pelos objetos presentes na Figura 3.5.

A extração de dados na ferramenta DEByE é embasada num modelo de objetos denominado DEByE OM (*DEByE Object Model*) [29]. O modelo DEByE OM é um modelo de dados para a representação de dados semi-estrututurados comumente encontrados em páginas da Web. Utilizando-se de construtores como *átomos*, *tuplas*, *listas* e *variantes*, o modelo é flexível o suficiente para representar a natureza semi-estruturada dos dados provenientes

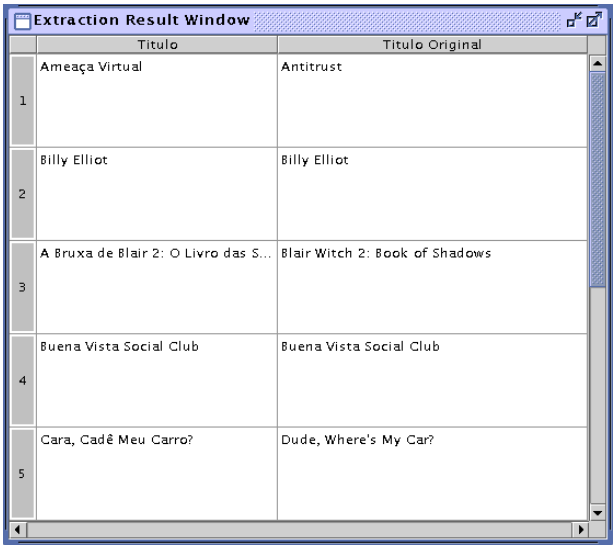


Figura 3.5: Resultado parcial da extração após a sessão de especificação de exemplos mostrada na Figura 3.4.

da Web. Os objetos resultantes do processo de extração são armazenados em arquivos XML, seguindo o formato DTORF (*DEByE Textual Object Repository Format*) [29], que é uma implementação XML do modelo de dados DEByE OM. Com a utilização do DTORF, os objetos extraídos podem ser posteriormente manipulados através de bibliotecas disponíveis para manipulação de arquivos XML, como, por exemplo, DOM (*Document Object Model*) [41].

Além de utilizar um modelo de dados flexível, a ferramenta DEByE implementa algoritmos para extração de dados flexíveis o suficiente para lhe permitir lidar com situações onde as páginas da Web podem apresentar objetos faltando atributos ou mesmo objetos com atributos fora de ordem [39].

3.3 A Ferramenta WebView

A ferramenta WebView [3, 4] permite a criação e a manutenção de visões de fontes de dados da Web. O usuário define como os dados extraídos de diferentes fontes devem ser integrados e atualizados em relação às fontes originais. A ferramenta também provê uma interface de alto nível baseada numa metáfora de grafos, onde os nodos representam as fontes de dados e as arestas representam os relacionamentos entre elas. A ferramenta WebView é composta por dois módulos principais, que serão descritos a seguir.

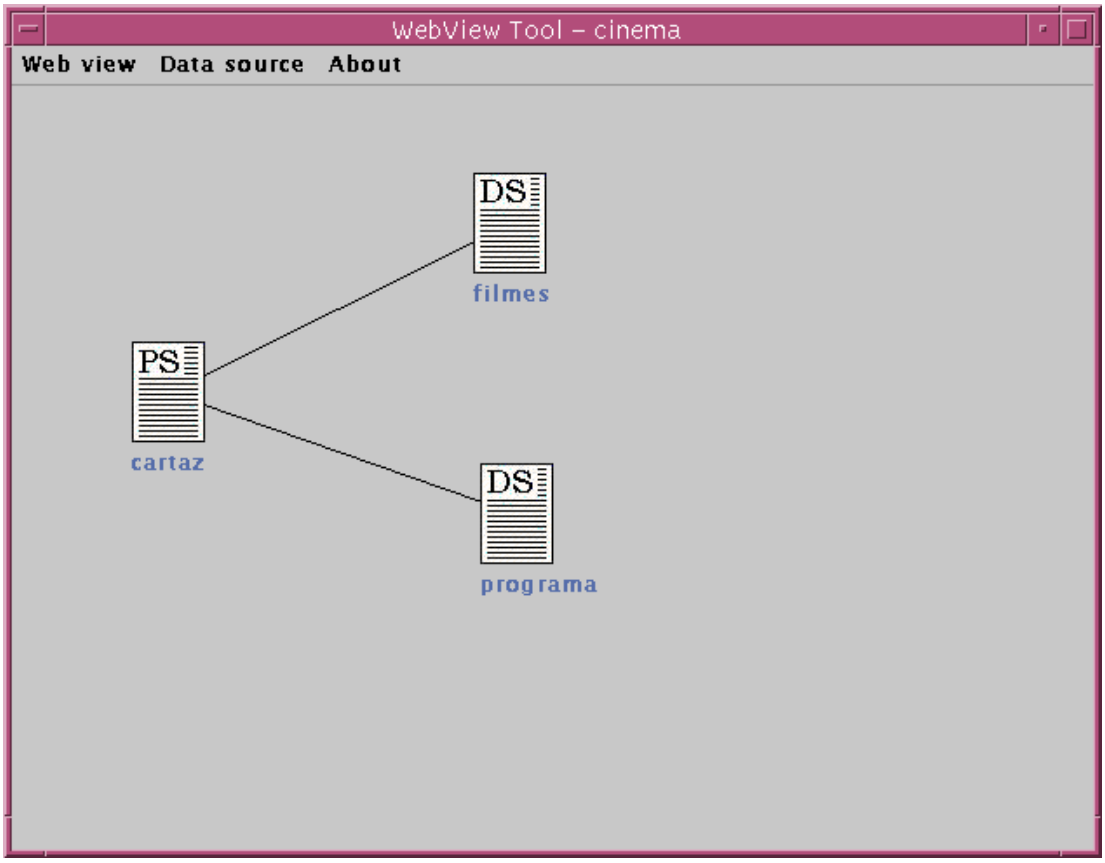


Figura 3.6: Definição de uma visão com dados relativos à programação de cinema em Belo Horizonte utilizando a ferramenta WebView.

A interface gráfica. Após a geração dos agentes de coleta pela ferramenta ASByE e dos extratores de dados pela ferramenta DEByE, a interface gráfica da ferramenta WebView é utilizada para a definição da visão. Basicamente, o usuário define quando e em que ordem os agentes ASByE devem ser executados para que possam alimentar os extratores DEByE. Os dados assim extraídos serão utilizados para materializar e atualizar a visão quando necessário. O usuário também define como e com que freqüência os dados de cada fonte de dados serão atualizados na visão. Existem algumas políticas de atualização que podem ser utilizadas para cada uma das fontes de dados que compõem uma visão, dependendo das suas características. As políticas de atualização das fontes de dados permitidas pela ferramenta WebView serão descritas no Capítulo 4.

O processador de visões. A saída da interface gráfica da ferramenta é o plano de materialização da visão, como pode ser visto pela arquitetura do ambiente na Figura 3.1.

O plano de materialização de uma visão é um arquivo XML que codifica as diretivas necessárias ao processador de visões na execução das tarefas de materialização e manutenção da visão. Essas diretivas são compostas pelos requisitos da visão, fornecidos pelo usuário quando a visão é definida utilizando a interface gráfica (Figura 3.1).

A Figura 3.6 ilustra a interface gráfica da ferramenta WebView na definição da visão utilizada no exemplo da Seção 1.2. É uma visão que contém dados relativos à programação de cinema em Belo Horizonte, retirados dos *sites* Guia de Cinema Online[4] e Estado de Minas[5].

A ferramenta WebView constitui o módulo da arquitetura do ambiente proposto (Figura 3.1) desenvolvido neste trabalho. Portanto, o Capítulo 4 é destinado a uma discussão mais detalhada da ferramenta. São abordadas em maiores detalhes as características da ferramenta, como as políticas de atualização das fontes de dados, o formato do plano de materialização de uma visão e como o plano é executado pelo processador de visões. Também são apresentados aspectos de implementação adotados no desenvolvimento da ferramenta.

Como foi discutido neste capítulo, o ambiente proposto para materialização e manutenção de visões de fontes de dados da Web é composto por ferramentas que provêm interfaces de alto nível. Estas interfaces tornam o ambiente mais adequado para a utilização por usuários menos especializados e para o rápido desenvolvimento de sistemas de informação baseados na Web, confirmando o que foi discutido nas Seções 1.3 e 1.4.

[4]Guia de Cinema Online - Tudo sobre os filmes em cartaz. http://www.guiadecinema.com.br
[5]Cinemas. http://cinema.uai.com.br

Capítulo 4

A Ferramenta WebView

Dos módulos componentes da arquitetura descrita no Capítulo 3 (Figura 3.1), o módulo relacionado à ferramenta WebView é o tema central deste trabalho. Neste capítulo as suas características são descritas em maiores detalhes. Em primeiro lugar, é descrita a sua interface gráfica através da qual o usuário pode criar visões de fontes de dados da Web. Através dessa interface, o usuário especifica basicamente quais serão as fontes de dados que constituirão a visão e como os dados extraídos de cada uma dessas fontes serão integrados. Essa permite ainda ao usuário especificar parâmetros que definirão como e com que freqüência as visões serão atualizadas. Em seguida, é descrito o processador de visões, que é o módulo da ferramenta que executa os agentes de coleta e extratores de cada fonte de dados com o objetivo de materializar e manter a visão consistente com as fontes de dados correspondentes. Após a descrição dos dois módulos da ferramenta WebView, são apresentados exemplos de visões criadas através da ferramenta, além de aspectos envolvidos na sua implementação.

4.1 A Interface Gráfica

Como foi inicialmente discutido no Capítulo 3, um dos módulos componentes da ferramenta WebView é a sua interface gráfica. Após a criação dos agentes de coleta e extratores de dados para cada uma das fontes que constituirão a visão, o próximo passo é a sua definição, utilizando a interface gráfica da ferramenta WebView. Através dessa interface, o usuário basicamente especifica quando e em que ordem os agentes de coleta devem ser executados, a fim de coletar as páginas das fontes de dados que alimentarão os extratores.

Além disso, o usuário define ainda como os dados extraídos de cada fonte serão integrados e atualizados em relação às fontes originais.

A ferramenta provê uma interface de alto nível baseada numa metáfora de grafos. Cada um dos nodos representa uma fonte de dados e as arestas entre eles representam os relacionamentos entre essas fontes.

A Figura 4.1 ilustra a interface gráfica da ferramenta com um nodo que representa uma fonte de dados. Para cada visão, é definido um grafo $G = (N, A)$, onde N é o conjunto de nodos do grafo e A é o conjunto de arestas.

Figura 4.1: Interface gráfica da ferramenta WebView com um nodo que representa uma fonte de dados.

Para que as fontes de dados pudessem ser representadas na interface gráfica da ferramenta WebView, foram desenvolvidos nodos diferentes para cada tipo de fonte de dados, de acordo com a categorização das fontes Suponhamos, por exemplo, que um usuário da Web esteja interessado em procurar um CD que deseja comprar em várias lojas virtuais distintas. Ele inicia sua busca pela informação desejada nos *sites* dessas lojas. Nesse momento, baseando-se na reunião de toda a informação encontrada nessas fontes de dados, ele passa a procurar informações complementares, novamente em uma ou mais fontes de dados distintas. Por exemplo, pode-se procurar informação discográfica sobre cada um dos CD's encontrados nas lojas virtuais. Novamente, pode-se procurar informações complementares

sobre a informação nessas fontes de dados, e assim por diante. Seria suficiente, portanto, a criação de uma visão que contenha reunida a informação proveniente de todas essas fontes de dados nas quais o usuário da Web procura a informação desejada. Os tipos de fonte de dados suportados pela ferramenta procuram se adaptar à maneira com que cada fonte é vista pelo usuário da Web.

Os seguintes tipos de fonte de dados estão disponíveis na ferramenta WebView:

- **Fonte de Dados Primária.** Esse é o tipo de fonte de dados que proverá os dados básicos para a visão, dentro do contexto específico da informação desejada. São as fontes de dados onde o usuário da Web inicia sua busca. Uma fonte de dados desse tipo pode ser usada para representar, por exemplo, *sites* de livrarias virtuais, em uma visão que contenha dados sobre informação bibliográfica.

- **Fonte de Dados Dependente.** Fontes de dados desse tipo apresentam dados complementares em relação a dados encontrados em outra fonte de dados, denominada *fonte de dados mestre*. Quando determinada informação é encontrada na fonte de dados mestre, informação complementar é coletada e extraída da *fonte de dados dependente*. Numa visão que contenha dados sobre informação bibliográfica, por exemplo, fontes de dados desse tipo podem representar *sites* que contenham informação sobre livros ou autores encontrados nos *sites* das livrarias virtuais.

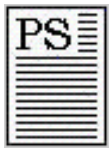
- **Fonte de Dados de União.** Em algumas situações, fontes de dados distintas em uma visão podem prover informação semanticamente relacionada. Isso ocorre quando se deseja obter informação espalhada por fontes de dados similares. Nesse caso, uma fonte de dados de união pode ser definida para representar a união do conteúdo dessas fontes. Ainda considerando como exemplo a visão sobre informação bibliográfica, uma fonte de dados de união pode ser usada para representar os dados extraídos de diversas livrarias virtuais distintas.

  Podem existir dois tipos de fonte de dados de união: (1) uma união de fontes de dados primárias e (2) uma união de fontes de dados dependentes. Uma mesma fonte de dados de união não pode envolver fontes de dados primárias e dependentes. Essa restrição se deve ao fato de que a existência da fonte de dados dependente implica na existência de uma operação de junção entre o seu conteúdo e o conteúdo da sua fonte de dados mestre, o que não acontece para a fonte de dados primária.

  Além disso, deve ser obedecido o requisito de que as fontes de dados envolvidas numa fonte de dados de união devem ser compatíveis de união [15]. Essa exigência pode ser

satisfeita através da modelagem adequada das fontes de dados, quando da criação dos extratores utilizando a ferramenta DEByE [39]. As características de modelagem da ferramenta [29] permitem que fontes de dados similares, que apresentem conteúdos semanticamente iguais, sejam modelados de forma a se tornarem compatíveis de união.

A Figura 4.2 apresenta os ícones utilizados para representar os nodos das fontes de dados primárias, dependentes e de união na interface gráfica da ferramenta WebView.



(a) Fonte de dados primária.

(b) Fonte de dados dependente.

(c) Fonte de dados de união.

Figura 4.2: Ícones utilizados para representar os nodos das fontes de dados primárias, dependentes e de união na interface gráfica da ferramenta WebView.

Durante a definição de uma visão, as fontes de dados que a compõem são definidas e o respectivo nodo é adicionado ao conjunto N de nodos do grafo de definição G. Além disso, o ícone que representa esse nodo é inserido na área de trabalho da interface gráfica. Após a inserção do nodo que representa uma fonte de dados, as propriedades desta fonte podem ser definidas através de operações que podem ser executadas sobre o nodo na interface. Essas operações podem ser acessadas através da utilização de um *menu popup* sobre cada um dos nodos definidos ou através dos menus da interface. As operações disponíveis são:

- **Definição das propriedades da fonte de dados.** Ao inserir uma fonte de dados na visão, apenas o ícone que representa o seu nodo é inserido na área de trabalho da interface. Nesse momento, o respectivo ícone aparece na área de trabalho da interface como um nodo isolado no grafo.

  No entanto, diversas informações sobre cada fonte de dados são necessárias para que a ferramenta WebView possa materializar e atualizar a visão corretamente. Através

da operação de definição das propriedades da fonte de dados, o usuário informa à ferramenta quais são essas informações. As informações necessárias para cada fonte de dados serão vistas mais adiante. Ao se definir as propriedades de uma fonte de dados f, é definido o seu relacionamento com as outras fontes presentes na visão. Dessa forma, o seu nodo n_f no grafo de definição G deixa de ser um nodo isolado, e G pode sofrer as seguintes alterações, estabelecendo o relacionamento entre f e as demais fontes de dados:

(a) Se f é uma fonte de dados primária, G permanece inalterado;

(b) Se f é uma fonte de dados dependente, é definida a fonte de dados mestre f_M de f. Caso a fonte de dados f_M já tenha exatamente uma fonte dependente f_D com o mesmo esquema de f, definido pelo seu extrator, então uma fonte de dados de união f_U é criada para representar a união de f e f_D. f_U é então uma união de fontes de dados dependentes. A fonte de dados f_U passa agora a ser dependente da fonte f_M. Seja $a = (n_{f_D}, n_{f_M})$ a aresta entre os nodos das fontes f_D e f_M. O grafo $G = (N, A)$ se torna então o grafo $G' = (N \cup \{n_{f_U}\}, (A - \{a\}) \cup \{(n_{f_D}, n_{f_U}), (n_f, n_{f_U}), (n_{f_M}, n_{f_U})\})$. Caso a fonte de dados de união f_U já exista, unindo fontes de dados dependentes de f_M com o mesmo esquema de f, então a fonte de dados f é apenas adicionada à união representada por f_U. O grafo $G = (N, A)$ se torna então o grafo $G' = (N, A \cup \{(n_{f_U}, n_f)\})$. Numa última situação, f é a primeira fonte dependente de f_M com o seu esquema. O grafo $G = (N, A)$ se torna então o grafo $G' = (N, A \cup \{(n_{f_M}, n_f)\})$.

(c) Se f é uma fonte de dados de união, é definido o conjunto de fontes $U = \{f_1, f_2, \ldots, f_n\}$, $n \geq 2$, que participam da união representada por f. Seja $A' = \{(n_{f_1}, n_f), (n_{f_2}, n_f), \ldots, (n_{f_n}, n_f)\}$ o conjunto das arestas que ligam cada fonte de dados $f_i \in U$ ao nodo da fonte f. O grafo $G = (N, A)$ se torna então o grafo $G' = (N, A \cup A')$.

- **Remoção de uma fonte de dados da visão.** A ferramenta ainda apresenta a flexibilidade de permitir que fontes de dados possam ser removidas durante a definição da visão.

4.1.1 As Formas de Integração das Fontes de Dados

Após a definição de todas as fontes de dados que constituem uma visão, assim como da definição de suas propriedades, o grafo de definição G correspondente determinará o esquema final da visão. O esquema final da visão é determinado de acordo com a forma com que os nodos no grafo se relacionam entre si. Uma aresta ligando o nodo de uma fonte de dados dependente ao nodo de sua fonte de dados mestre representa uma operação de junção entre as duas fontes, mais especificamente, uma operação de junção à esquerda (*left join*) [15], como será visto na Seção 4.2.

O grafo resultante da definição de uma visão tem algumas caraterísticas particulares. Os nodos e arestas não são inseridos aleatoriamente, o que resultaria num grafo em que nodos representando fontes de dados primárias, dependentes e de união estivessem conectados desorganizadamente. A forma com que eles estão conectados depende de como as propriedades de cada uma das fontes de dados foram definidas. Dessa forma, a disposição dos nodos e arestas no grafo resultante é que determina a forma com que os dados extraídos de cada uma das fontes serão integrados quando a visão for materializada.

Pode-se agora descrever, através de operações de junção e de união sobre as fontes de dados, como o esquema final da visão é definido a partir do grafo de definição G. De forma bem simplificada, uma operação de junção em uma visão é representada por uma aresta no grafo correspondente entre o nodo de uma fonte de dados dependente e o nodo de sua fonte de dados mestre; e uma operação de união é representada por uma fonte de dados de união e as arestas conectando o seu nodo aos nodos das fontes de dados que dela participam. Mais formalmente, pode-se definir essas operações pelas seguintes expressões:

(a) A conexão entre uma fonte de dados de união f_U e cada uma das fontes f_i pertencentes ao conjunto de fontes de dados $U = \{f_1, f_2, \ldots, f_n\}$, $n \geq 2$, que dela participam define a operação de união de acordo com a expressão:

$$f_1 \cup f_2 \cup \ldots \cup f_n \tag{4.1}$$

(b) A conexão entre uma fonte de dados f e cada uma das fontes f_i pertencentes ao conjunto de suas fontes de dados dependentes $D = \{f_1, f_2, \ldots, f_n\}$ define a operação de junção de acordo com a expressão:

$$(f \overrightarrow{\bowtie} f_1) \bowtie (f \overrightarrow{\bowtie} f_2) \bowtie \ldots \bowtie (f \overrightarrow{\bowtie} f_n) \tag{4.2}$$

Os atributos das fontes de dados envolvidas sobre os quais a junção será feita são definidos pelo usuário quando as propriedades da fonte de dados dependente são definidas, o que será visto a seguir.

4.1.2 Informação Necessária sobre as Fontes de Dados

Como já foi discutido, após a inserção de uma fonte de dados na visão, propriedades sobre essa fonte devem ser definidas. São essas propriedades que vão determinar a disposição dos nodos e arestas no grafo resultante, uma vez que elas determinam como as fontes de dados se relacionam entre si.

Essas informações são necessárias para a correta materialização e manutenção da visão, pois descrevem o comportamento de cada uma das fontes de dados. Esse comportamento determina, em linhas gerais, como os dados de cada fonte sofrem alterações e, conseqüentemente, como devem ser atualizados na visão.

As propriedades de uma fonte de dados são específicas para cada tipo de fonte, ou seja, depende da sua natureza (primária, dependente ou de união). Para uma fonte de dados primária, essas informações incluem:

• **Nome da fonte de dados.** O nome da fonte de dados na visão. É também o nome do arquivo XML que armazena os dados extraídos da fonte.

• **Agente de coleta de páginas.** O agente de coleta gerado pela ferramenta ASByE [22, 23] responsável por coletar as páginas desta fonte de dados.

• **Extrator de dados.** O extrator de dados gerado pela ferramenta DEByE [29, 39] reponsável por extrair os dados das páginas coletadas.

• **Política de atualização.** A política de atualização para uma fonte de dados determina como e com que freqüência os dados dessa fonte são atualizados. Essa forma de atualização depende diretamente da natureza da fonte e do comportamento dos seus dados, em relação às suas modificações. Para uma fonte de dados primária, as seguintes políticas de atualização estão disponíveis:

(a) **Polling:** Na política do tipo *polling*, a fonte de dados é continuamente monitorada para a verificação de alterações em seu conteúdo. Dessa forma, novos dados encontrados na fonte são inseridos na visão e dados que não estão mais presentes na fonte são removidos da visão.

Quando a fonte de dados é atualizada por *polling*, o usuário define parâmetros temporais que determinam com que freqüência a fonte de dados será monitorada. Esses

parâmetros podem ser fixos ou ainda determinados calculando-se a freqüência média com que a fonte de dados sofre alterações.

(b) **Pushing:** Nessa política, a fonte de dados se encarrega de informar à ferramenta WebView quando seus dados sofrem alterações. Esse paradigma para divulgação da informação pela Web tem sido chamado de *Webcasting* [19] na literatura recente. Diversos *sites* encontrados na Web oferecem serviços que enviam periodicamente aos usuários mensagens com novos produtos, catálogos ou notícias, por exemplo. Dessa forma, a ferramenta monitora a alteração no conteúdo de um arquivo externo alterado pela fonte, que efetivamente contém os dados a serem extraídos, como, por exemplo, a caixa de correio do usuário.

No caso de uma fonte de dados atualizada por *pushing*, o usuário define o arquivo a ser monitorado, que conterá os dados enviados pela fonte. Alterações no conteúdo desse arquivo são consideradas alterações na fonte de dados.

(c) **Sob demanda:** Uma fonte de dados pode também ser atualizada sob demanda, quando se tornar necessário.

• **Atributos identificadores.** Quando o extrator para uma fonte de dados f é construído, utilizando a ferramenta DEByE [39], é definido um conjunto de atributos $A = \{a_1, a_2, \ldots, a_n\}$ para os dados presentes na fonte. Esse conjunto de atributos define o esquema da fonte de dados f. Os tipos que cada um dos atributos a_i pode assumir, são derivados dos construtores *átomos*, *tuplas*, *listas* e *variantes* permitidos pela ferramenta DEByE. Na fonte de dados da Figura 1.4, por exemplo, os dados presentes apresentam os atributos Titulo, Diretor, Roteiro, Elenco, Genero e Duracao, onde o atributo Elenco é um atributo do tipo *lista* e os demais são atributos do tipo *átomo*.

Dentre os atributos $a_i \in A$ definidos para uma fonte de dados, o usuário define um subconjunto $I \subseteq A$ de atributos atômicos que serão considerados os atributos identificadores da fonte de dados. Tais atributos têm uma função similar às chaves primárias no modelo relacional [15]. Os atributos $a_i \in I$ definidos como os atributos identificadores são utilizados para a comparação dos dados presentes na visão e os dados presentes na fonte de dados. Dessa forma, pode-se determinar quais são os dados na fonte a serem inseridos na visão e quais são os dados na visão a serem removidos, por não estarem mais presentes na fonte. A necessidade de um conjunto de atributos identificadores será discutida em maiores detalhes na Seção 4.2.

A Figura 4.3 mostra como são definidos o nome da fonte de dados, além de seu agente de coleta e extrator de dados. A Figura 4.4 ilustra como são definidas as propriedades adicionais para uma fonte de dados primária através da interface da ferramenta WebView.

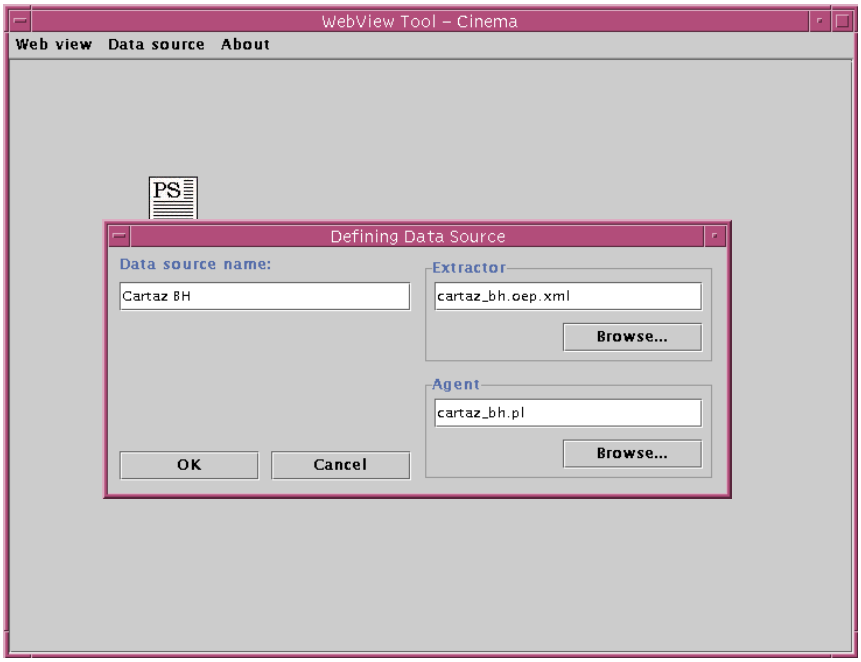


Figura 4.3: Definição do nome de uma fonte de dados, além de seu agente de coleta e extrator de dados.

Para uma fonte de dados dependente, por sua vez, o usuário define, além do nome da fonte de dados, agente de coleta e extrator de dados (Figura 4.3), as seguintes propriedades que a descrevem, necessárias para sua materialização e manutenção:

• **Política de atualização.** Além das políticas *polling* e *pushing* permitidas para fontes de dados primárias, as fontes de dados dependentes suportam ainda uma política de atualização adicional, chamada *triggering*. Essa política determina que os dados provenientes de uma fonte de dados dependente são estáticos em relação à sua fonte de dados mestre, pois são modificados apenas quando os dados da fonte de dados mestre também são. Assim, dados dessa fonte apenas são inseridos na visão quando o dado correspondente é encontrado na fonte mestre e removidos da visão quando o dado correspondente não está mais presente na fonte de dados mestre.

• **Atributos identificadores.** Da mesma forma que para uma fonte de dados primária, dentre o conjunto de atributos $A = \{a_1, a_2, \ldots, a_n\}$ definidos para a fonte de dados quando

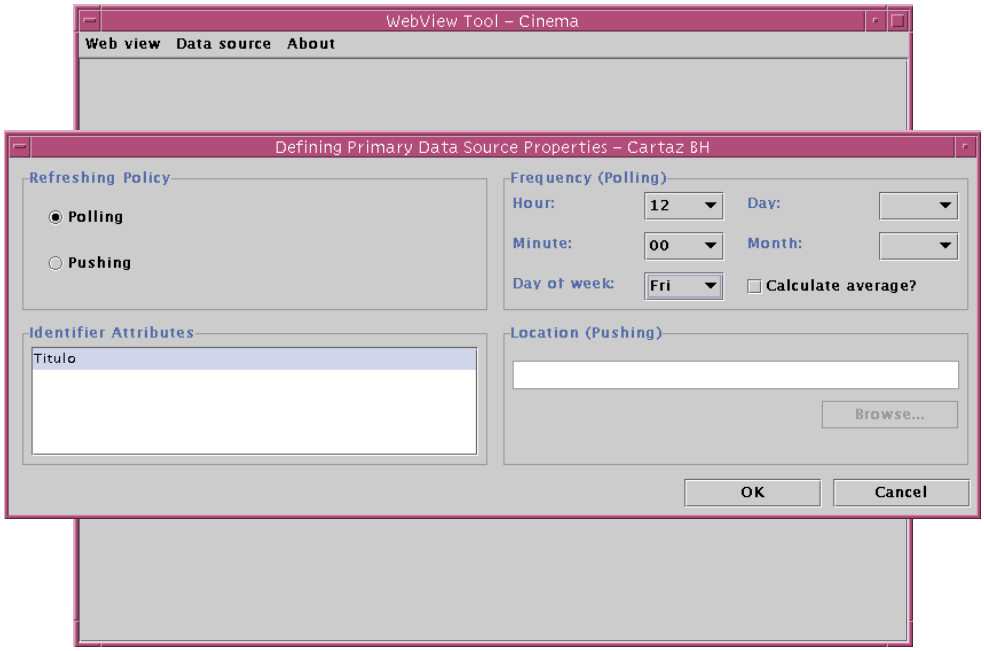


Figura 4.4: Definição das propriedades adicionais para uma fonte de dados primária.

o seu extrator foi construído, um subconjunto $I \subseteq A$ de atributos atômicos é definido como o conjunto de atributos identificadores para a fonte de dados.

- **Fonte de dados mestre.** Uma das informações necessárias sobre uma fonte de dados dependente f_D é a definição da sua fonte de dados mestre f_M, para a qual ela provê dados complementares. No entanto, nem todas as demais fontes de dados podem ser definidas como uma fonte de dados mestre. Algumas restrições são impostas sobre os possíveis candidatos a fonte de dados mestre de f_D. Uma fonte de dados f_M apenas pode ser definida como a fonte de dados mestre de f_D, se f_M não participa de nenhuma fonte de dados de união f_U, onde f_U é uma fonte de dados de união qualquer. Essa restrição impede que a fonte de dados f_M tenha f_D como fonte de dados dependente, o que implicaria na existência de uma operação de junção entre f_M e f_D. Essa operação de junção eliminaria a compatibilidade de união imposta pela existência da fonte de dados de união f_U.

- **Atributos de junção.** A última propriedade necessária a ser definida para uma fonte de dados dependente f_D são os atributos de junção com a sua fonte de dados mestre f_M. Sejam $A_M = \{a_{M1}, a_{M2}, \ldots, a_{Mn}\}$ o conjunto de atributos da fonte de dados mestre f_M e $A_D = \{a_{D1}, a_{D2}, \ldots, a_{Dm}\}$ o conjunto de atributos da fonte de dados dependente f_D. Os atributos de junção entre as duas fontes de dados são definidos como um conjunto de pares ordenados $J \subseteq A_M \times A_D$. Em cada par ordenado (a_{Mi}, a_{Di}), a_{Mi} e a_{Di} são atributos

atômicos das fontes de dados f_M e f_D, respectivamente. A Figura 4.6 ilustra como os atributos de junção de uma fonte de dados dependente são definidos através da interface da ferramenta WebView.

A Figura 4.5 ilustra como são definidas as propriedades adicionais para uma fonte de dados dependente através da interface da ferramenta WebView. Pode-se observar ainda, através do item Use join attributes for fetching? na Figura 4.5, que a coleta de páginas na fonte de dados f é determinada pelos atributos de junção com a sua fonte de dados mestre. Em casos como esse, uma página distinta deve ser coletada na fonte de dados dependente para cada objeto da fonte de dados mestre. Isso ocorre, por exemplo, através do preenchimento de formulários HTML ou na seqüência de um *hyperlink* parametrizado na fonte de dados dependente utilizando atributos do objeto da fonte de dados mestre. Essa funcionalidade é suportada pelos agentes de coleta gerados pela ferramenta ASByE [22, 23], como foi visto na Seção 3.1 e será mostrado com mais detalhes na Seção 4.2.

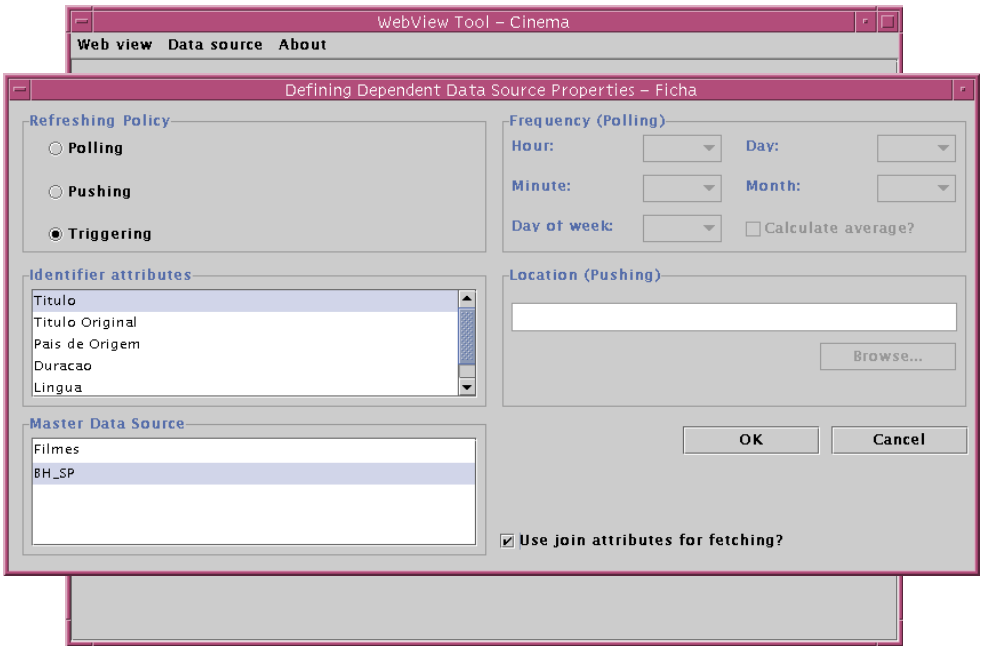


Figura 4.5: Definição das propriedades adicionais para uma fonte de dados dependente.

Finalmente, para uma fonte de dados de união, as únicas propriedades a serem definidas são o seu nome e o conjunto de fontes de dados $U = \{f_1, f_2, \ldots, f_n\}$, $n \geq 2$, que dela participam. A Figura 4.7 mostra como essas propriedades são definidas na interface gráfica da ferramenta.

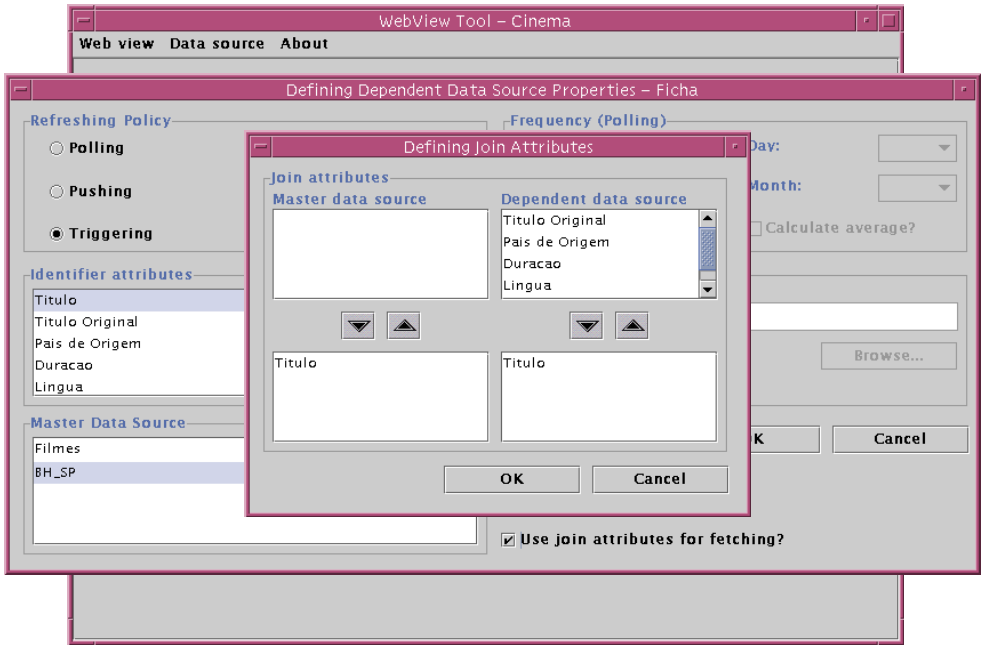


Figura 4.6: Definição dos atributos de junção entre fontes de dados mestre e dependente.

Após a definição de todas as fontes de dados que constituem a visão e do relacionamento entre elas, o *plano de materialização da visão* pode ser gerado. Esse plano nada mais é do que um arquivo XML que codifica todas as propriedades definidas para cada fonte de dados. O formato desse plano e como ele é processado pelo processador de visões será o assunto da próxima seção.

4.2 O Processador de Visões

Definida a visão através da interface gráfica da ferramenta WebView, o plano de materialização da visão é então gerado. Nesta seção, é apresentado o processador de visões, que tem como entrada esse plano, saída da interface gráfica (Figura 3.1). São discutidos o formato do plano e como este é executado pelo processador de visões, perfazendo as ações necessárias para a materialização e manutenção de uma visão de fontes de dados da Web.

4.2.1 Formato do Plano de Materialização de uma Visão

Como já foi apresentado, o plano de materialização de uma visão é um arquivo XML que contém todas as propriedades que foram definidas para cada fonte de dados, quando

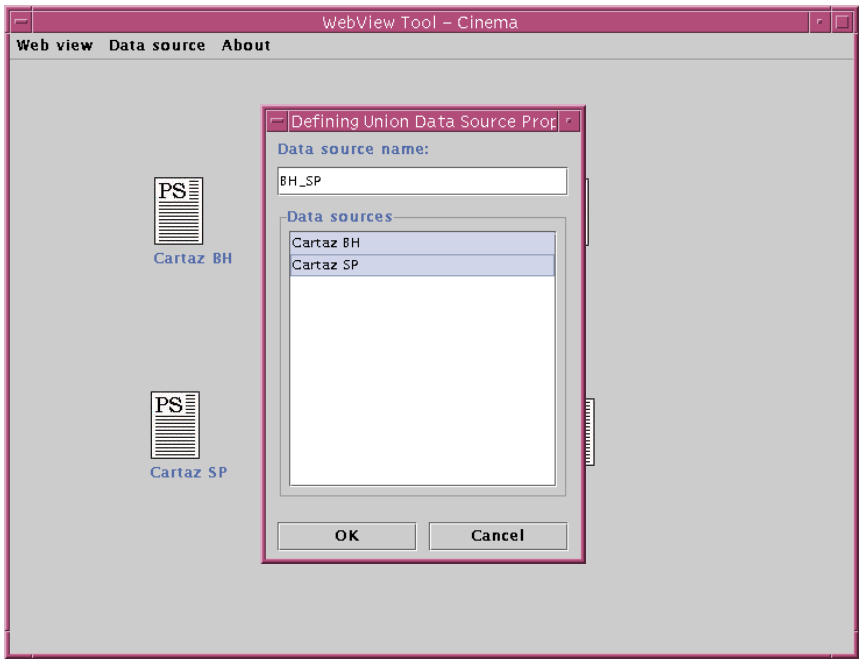


Figura 4.7: Definição das propriedades de uma fonte de dados de união.

a visão foi definida. O formato de um plano de materialização define a estrutura na qual o arquivo XML armazena as propriedades definidas para as fontes de dados. Além das propriedades das fontes de dados, o plano de materialização contém informação sobre a visão como um todo. Essa informação inclui: o nome da visão, que é também o nome do arquivo XML onde a visão será armazenada, e a estratégia de atualização da visão, que determina se a atualização da visão é incremental ou não. Em visões com manutenção incremental, novos dados encontrados nas fontes de dados primárias são inseridos na visão, e dados removidos das fontes de dados primárias são também removidos da visão. A atualização não incremental determina que todo o conteúdo de determinada fonte de dados primária e, conseqüentemente, de suas fontes de dados dependentes, seja removido da visão e rematerializado.

Essa informação relacionada à visão é incluída no plano de materialização nos atributos name e incremental, na *tag* WEBVIEW, que é a raiz do arquivo XML que armazena o plano. A Figura 4.8 mostra a estrutura de um arquivo XML utilizado para armazenar um plano de materialização de uma visão.

As propriedades de uma fonte de dados primária são definidas na *tag* PRIMARY. Essa *tag* apresenta os atributos identifiers e name, para a representação dos atributos identificadores e do nome da fonte de dados, respectivamente. Internamente a essa *tag* estão as propriedades

```
<WEBVIEW incremental="..." name="...">
    ...
    <PRIMARY identifiers="..." name="...">
        <REFRESH policy="...">
            <FREQUENCY calcavg="..." day="..." dayofweek="..." hour="..."
                       minute="..." month="..."></FREQUENCY>
            ou
            <LOCATION name="..."></LOCATION>
        </REFRESH>
        <FETCH agent="..."></FETCH>
        <EXTRACT extractor="..."></EXTRACT>
    </PRIMARY>
    ...
    <DEPENDENT master="..." name="..." identifiers="...">
        <REFRESH policy="...">
            <FREQUENCY calcavg="..." day="..." dayofweek="..." hour="..."
                       minute="..." month="..."></FREQUENCY>
            ou
            <LOCATION name="..."></LOCATION>
        </REFRESH>
        <FETCH agent="..." usejoinattributes="..."></FETCH>
        <EXTRACT extractor="..."></EXTRACT>
        <JOIN>
            <ATTRIBUTE dependent="..." master="..."></ATTRIBUTE>
        </JOIN>
    </DEPENDENT>
    ...
    <UNION name="..." master="...">
        <DATASOURCE name="..."></DATASOURCE>
        ...
        <DATASOURCE name="..."></DATASOURCE>
    </UNION>
    ...
</WEBVIEW>
```

Figura 4.8: Estrutura de um arquivo XML utilizado para armazenar um plano de materialização de uma visão.

específicas da fonte de dados, já descritas anteriormente. A *tag* REFRESH define a política de atualização da fonte de dados, no seu atributo policy. Dependendo da política de atualização, a *tag* pode ter como *tag* filha a *tag* FREQUENCY para atualização do tipo *polling*, ou a *tag* LOCATION para atualização do tipo *pushing*. Pode-se observar que os

atributos dessas *tags* contêm informação para cada política de atualização. Nas *tags* FETCH e EXTRACT são indicados os agentes de coleta e extratores de dados, respectivamente, para a fonte de dados.

Na *tag* DEPENDENT são definidas as propriedades de uma fonte de dados dependente. Além dos atributos name e identifiers, similares aos de uma fonte de dados primária, existe ainda o atributo master, que define o nome da fonte de dados mestre. Suas *tags* filhas REFRESH, FETCH e EXTRACT têm função similar à de uma fonte de dados primária. A *tag* FETCH, entretanto, tem um atributo adicional, usejoinattributes. Esse atributo determina se o agente de coleta deve ser executado uma vez para cada dado presente na sua fonte de dados mestre, ou se uma execução é suficiente para todos os dados. No exemplo discutido no Capítulo 1, a fonte de dados Programa (Figura 1.5) apresenta uma página distinta com os horários de exibição de cada filme em cartaz presente na fonte Cartaz (Figura 1.3). Dessa forma, usando os atributos de junção entre as duas fontes, nesse caso, o título do filme, o agente de coleta para a fonte de dados Programa executa uma busca independente para cada filme encontrado. Caso a fonte Programa contivesse os horários de exibição de todos os filmes numa única página, uma única execução do agente de coleta seria suficiente. Esse tipo de comportamento de algumas fontes de dados pode ser tratado devido a uma facilidade oferecida pela ferramenta ASByE, que é a geração de agentes de coleta parametrizados, conforme foi discutido na Seção 3.1. A *tag* JOIN, por sua vez, contém os atributos de junção entre a fonte de dados dependente e sua fonte de dados mestre.

Finalmente, a *tag* UNION define as propriedades de uma fonte de dados de união. Basicamente, é armazenado o nome de cada uma das fontes que participam da união, nas *tags* DATASOURCE. O atributo master na *tag* UNION informa qual a sua fonte de dados mestre, caso seja uma união de fontes de dados dependentes.

A Figura 4.8 apresenta partes de um plano de materialização que ilustram toda a sua possível estrutura, conforme descrito acima. Como em uma visão é obrigatória a existência de pelo menos uma fonte de dados primária, o plano de materialização de uma visão terá pelo menos uma *tag* PRIMARY. Portanto, é possível que as *tags* DEPENDENT e UNION não apareçam.

O ciclo de vida de uma visão envolve duas fases distintas: (1) a sua materialização, quando todas as suas fontes de dados são materializadas e integradas; e (2) a sua atualização, quando uma ou mais de suas fontes de dados são materializadas, e as suas alterações são propagadas para a visão. A execução de cada uma dessas fases na ferramenta WebView,

a cargo do processador de visões, será descrita nas próximas seções.

4.2.2 Materialização

Descreve-se agora com mais detalhes, como uma visão é materializada, através da execução ordenada dos agentes de coleta e extratores de dados das fontes que a constituem. A ordem de execução dos agentes e extratores é determinada pela forma com que o plano de materialização da visão é executado pelo processador de visões.

A materialização deve partir das fontes de dados primárias, das quais todas as outras fontes são dependentes. O primeiro passo então a ser tomado é a execução dos agentes de coleta de cada uma das fontes de dados primárias. Nesse ponto, uma questão importante, relacionada ao desempenho, deve ser observada. Como as fontes de dados primárias são independentes, os seus agentes de coleta são executados em paralelo, em processos isolados. Essa execução paralela de tarefas é importante, pois permite um melhor desempenho da ferramenta, ao contrário da execução seqüencial dos agentes.

À medida que as requisições feitas por cada agente de coleta às fonte de dados vão sendo respondidas, as páginas recebidas são repassadas aos respectivos extratores de dados. Com a execução dos extratores sobre as páginas coletadas, um repositório temporário é criado para armazenar os dados de cada fonte. Cada um desses repositórios nada mais é do que um arquivo XML no formato TOR (*Textual Object Repository*) [29] que implementa o modelo DEByE-OM, modelo de dados semi-estruturados da ferramenta DEByE.

Antes de prosseguir com a materialização das demais fontes de dados, uma ação importante deve ser executada sobre as fontes de dados primárias. Além dos atributos normalmente extraídos da fonte, um atributo especial é adicionado a todos os objetos de uma fonte de dados primária. Trata-se do atributo *Primary Data Source*, ou seja, o próprio nome da fonte de dados. A necessidade desse atributo decorre do fato de que uma mesma visão pode conter mais de uma fonte de dados primária, e em determinados momentos é necessário distinguir qual a fonte de dados primária é a origem do objeto.

Consideremos a existência de dois objetos o_1 e o_2, provenientes de fontes de dados primárias distintas, f_{P_1} e f_{P_2}, respectivamente. Suponhamos, agora, que o_1 e o_2 sejam exatamente iguais em um subconjunto de seus atributos, que inclui seus atributos identificadores. Se a fonte de dados f_{P_1} estiver sendo atualizada e o objeto o_1 tiver sido removido de f_{P_1}, o mesmo deve ser também removido da visão. Entretanto, sem o atributo no objeto que define qual fonte de dados primária é a sua origem, ambos os objetos o_1 e o_2 serão removidos da visão. A remoção de o_2 da visão é incorreta, uma vez que ele ainda está

presente na fonte de dados primária f_{P_2}.

Existe ainda outro problema relacionado à necessidade da existência do atributo *Primary Data Source*. Suponhamos agora que existam os mesmos objetos o_1 e o_2 provenientes das mesmas fontes de dados primárias f_{P_1} e f_{P_2}, e que essas fontes possuam as fontes de dados dependentes f_{D_1} e f_{D_2}, respectivamente. Os objetos o_1 e o_2 podem ter exatamente os mesmos valores para os atributos de junção entre as fontes primárias e dependentes. Na hipótese da fonte f_{D_1} estar sendo atualizada, e o objeto o_1 tiver sido removido, os respectivos atributos de o_1 devem se tornar nulos na visão, em virtude da operação de junção à esquerda. Sem o atributo identificador da fonte de dados primária, ambos os objetos o_1 e o_2 teriam seus atributos relacionados às fontes dependentes convertidos para valores nulos, o que também é incorreto. Logo, a necessidade de se identificar de qual fonte de dados primária os objetos o_1 e o_2 provêm.

Após a materialização de cada fonte de dados primária, as fontes de dados de união sobre elas definidas são materializadas. Seja f_U uma fonte de dados de união definida sobre um conjunto de fontes de dados primárias $U = \{f_{P_1}, f_{P_2}, \ldots, f_{P_n}\}$, $n \geq 2$. Os repositórios temporários de cada uma das fontes de dados f_{P_i} são então utilizados para a materialização da fonte de dados f_U. Um repositório temporário para a fonte f_U também é criado.

Uma vez materializadas as fontes de dados primárias e as fontes de dados de união sobre elas definidas, tem início um processo recursivo para a materialização das fontes de dados dependentes e a execução da operação de junção entre elas e suas fontes de dados mestre. Consideremos, por exemplo, f uma fonte de dados a ser materializada e $D = \{f_{D_1}, f_{D_2}, \ldots, f_{D_n}\}$ o conjunto das fontes dependentes de f. A fonte de dados f deve ser materializada antes de suas fontes dependentes, uma vez que a coleta nessas fontes pode depender dos dados já extraídos de f. Cada uma das fontes $f_{D_i} \in D$ é então materializada, causando também a materialização das fontes dependentes de f_{D_i}. Esse processo se repete até que uma das fontes materializadas não tenha mais dependentes. Após a materialização de todas as fontes de dados, tem início a execução da operação de junção entre fontes dependentes e mestres. A operação de junção é iniciada no sentido inverso ao da materialização, pelas últimas fontes materializadas, obedecendo à expressão definida em (4.2).

Como os objetos extraídos das fontes de dados estão de acordo com o modelo DEByE-OM, os seus atributos podem ter uma estrutura complexa definida pelos construtores *átomos*, *tuplas*, *listas* e *variantes*. Entretanto, apenas os atributos atômicos estão envolvidos na operação de junção entre duas fontes de dados. Além disso, a operação de junção

definida considera dois objetos de fontes de dados distintas como sendo a mesma entidade do mundo real se ambos têm exatamente o mesmo valor nos atributos de junção. Sabe-se que essa abordagem pode falhar em muitos casos e não é a ideal na manipulação de dados provenientes da Web. Uma abordagem alternativa seria considerar, não a igualdade entre os atributos de junção, mas sim uma aproximação entre eles baseada em sua similaridade textual, como proposto em [11].

Conforme já havia sido mencionado na Seção 4.1, a operação de junção entre fontes de dados dependentes e fontes de dados mestres é uma junção à esquerda (*left join*) [15]. Na operação de junção à esquerda entre duas fontes de dados f_1 e f_2 ($f_1 ⟕ f_2$), mesmo os dados de f_1 que não apresentam dados correspondentes na fonte f_2 fazem parte do resultado da operação. Nesse caso, os atributos da fonte f_2 no resultado assumem o valor nulo.

A princípio, não há nenhuma restrição para que seja a operação de junção comum. Entretanto, para aplicações que utilizam dados provenientes da Web, é interessante manter na visão dados incompletos, mostrando que dados podem ser encontrados em uma fonte de dados, sem que dados correspondentes sejam encontrados nas suas fontes dependentes. Esse problema é conseqüência do fato de que fontes de dados da Web envolvidas em uma mesma visão podem estar sob domínios diferentes, ou seja, administradas independentemente. Isso não garante que, embora as duas fontes possam tratar de informação de um mesmo contexto específico, o conteúdo de ambas esteja sincronizado.

A partir da materialização da visão, o processador de visões deve tomar as medidas necessárias para mantê-la consistente com as fontes de dados originais. Para isso, deve agendar a primeira atualização de cada uma das fontes de dados presentes na visão. Entretanto, somente a atualização das fontes de dados cuja política de atualização é do tipo *polling* são agendadas. Esse agendamento é feito em função dos parâmetros temporais definidos pelo usuário quando as propriedades da fonte foram definidas, na definição da visão (Figura 4.5), utilizando o comando *at* do sistema operacional *Unix*. As fontes de dados cuja política de atualização é do tipo *pushing*, têm suas próximas atualizações quando informadas pela fonte de dados; enquanto que as fontes atualizadas por *triggering* são atualizadas quando a sua fonte de dados mestre também for.

4.2.3 Atualização

A forma como uma visão é atualizada, em geral, depende de como as suas fontes de dados são atualizadas, ou seja, depende da política de atualização de cada uma delas.

As fontes de dados envolvidas em uma visão não são atualizadas simultaneamente, ou seja, a partir do momento em que a visão é materializada, a atualização de cada uma é independente das demais. O que será descrito agora é então quando e como o processador de visões determina a atualização de cada fonte de dados.

Durante a atualização de uma fonte de dados, um processo similar ao da materialização ocorre: o agente de coleta para a fonte de dados é executado e as páginas coletadas alimentam o extrator, produzindo um conjunto de objetos que serão utilizados para atualizar a visão. Basicamente, o trabalho de atualizar uma visão consiste em remover os objetos que não estão mais presentes nas fontes de dados e inserir os novos objetos encontrados nas fontes. Entretanto, alguns detalhes estão envolvidos nesse processo, distinguindo a forma com que uma fonte de dados primária e uma fonte de dados dependente são atualizadas. Uma fonte de dados de união não é atualizada diretamente, uma vez que as fontes de dados que a constituem podem apresentar políticas diferentes, o que as levaria a serem atualizadas em momentos distintos.

Atualizando uma Fonte de Dados Primária

Para a atualização de uma fonte de dados primária f_P, a primeira ação a ser executada pelo processador de visões é a coleta das páginas de f_P utilizando o agente de coleta $a(f_P)$, definido para a fonte. Após a coleta das páginas, o extrator de dados $x(f_P)$ é utilizado para a extração dos dados das páginas coletadas. Para formalizar a atualização de uma fonte de dados em uma visão, utilizaremos a definição a seguir.

Definição 1 Seja o um objeto de uma fonte de dados f qualquer e $A = \{a_1, a_2, \ldots, a_n\}$ o conjunto dos atributos definidos para f. Define-se a projeção de o sobre um conjunto de atributos $A' \subseteq A$, denotada por $o[A']$, como sendo o subobjeto de o que apresenta apenas os atributos presentes em A'.

Uma vez que o conjunto de objetos presentes na fonte de dados foi extraído, esse conjunto é utilizado para a atualização da visão. Seja $O = \{o_1, o_2, \ldots, o_n\}$ o conjunto de objetos extraídos de f_P e $V = \{v_1, v_2, \ldots, v_m\}$ o conjunto de objetos já armazenados na visão cujo atributo *Primary Data Source* tem como valor o nome da fonte de dados f_P. O conjunto V contém apenas os objetos da visão que são provenientes da fonte de dados f_P. Seja ainda o conjunto A_I dos atributos identificadores da fonte de dados f_P. Considerando-se os dois conjuntos distintos de objetos, os seguintes cenários são possíveis

para a atualização da fonte f_P. Como conseqüência de cada um desses cenários, ações específicas são executadas pelo processador de visões para a atualização da visão.

(a) Existe um subconjunto $O' = \{o_1, o_2, \ldots, o_k\}$, $O' \subseteq O$ dos objetos extraídos, tal que $\nexists v_i \in V$, $1 \leq i \leq m$, tal que $v_i[A_I] = o_j[A_I]$, $\forall o_j \in O'$. Os objetos do conjunto O' são tais que foram extraídos da fonte f_P e ainda não estão presentes na visão, portanto, devem ser inseridos.

O conjunto de objetos O' por si só é considerado como uma fonte de dados f qualquer. Como os objetos de O' são originários de f_P, a fonte f deve ser integrada com as fontes de dados dependentes de f_P como se fosse a própria fonte f_P, seguindo o processo recursivo descrito na Seção 4.2.2. Os objetos resultantes dessa integração de f com as fontes dependentes de f_P são então inseridos na visão.

(b) Existe um subconjunto $V' = \{v_1, v_2, \ldots, v_k\}$, $V' \subseteq V$ dos objetos da visão, tal que $\nexists o_i \in O$, $1 \leq i \leq n$, tal que $o_i[A_I] = v_j[A_I]$, $\forall v_j \in V'$. Os objetos do conjunto V' são tais que estavam presentes na visão e não estão mais presentes na fonte de dados, portanto, devem ser removidos.

Como o conjunto V' é constituído de objetos que não mais existem na fonte de dados f_P, esses objetos são simplesmente removidos da visão.

(c) Existe um subconjunto $O' = \{o_1, o_2, \ldots, o_k\}$, $O' \subseteq O$ dos objetos extraídos, tal que $\forall o_i \in O'$, $\exists v_j \in V$, tal que $v_j[A_I] = o_i[A_I]$. Os objetos do conjunto O' são tais que foram extraídos de f_P e já estavam presentes na visão. Seja A_J o conjunto dos atributos de junção entre a fonte f_P e uma de suas fontes de dados dependentes f_D qualquer. Nesse caso, duas situações distintas podem ocorrer para cada objeto $o_i \in O'$:

(1) $o_i[A_J] \neq v_j[A_J]$. O objeto o_i da fonte de dados f_P não satisfaz mais os critérios de junção entre as duas fontes de dados f_P e f_D.

Nessa situação, um novo processo de coleta e extração deve ser feito na fonte de dados f_D, e a operação de junção com a fonte de dados f_P é refeita. Entretanto, a execução dessa operação pode tornar a operação de junção entre a fonte de dados f_D e alguma de suas fontes dependentes inconsistente, caso o valor dos atributos de junção entre f_D e alguma de suas fontes dependentes tenha mudado no objeto v_j, após a junção ser refeita.

Novamente, um novo processo de coleta e extração deve ser feito na fonte de dados dependente de f_D, e a operação de junção entre as duas fontes é refeita, até que o objeto permaneça com todas as operações de junção consistentes. Se em algum ponto da coleta dos objetos nas fontes dependentes nenhum objeto é extraído, os atributos relacionados a essas fontes de dados são convertidos para o valor nulo.

(2) $o_i[A_J] = v_j[A_J]$. O objeto o_i da fonte de dados f_P ainda satisfaz os critérios de junção entre as fontes de dados f_P e f_D.

Nesse caso, nenhuma alteração é feita nos objetos de V.

Para cada um dos cenários descritos acima, uma ação diferente deve ser tomada para a correta atualização do conteúdo da fonte de dados na visão. Pode-se observar que esses cenários são independentes da política de atualização da fonte de dados f_P. A política de atualização apenas determina quando a fonte deve ser atualizada e de que forma as alterações em seu conteúdo devem ser monitoradas.

Finalmente, deve ser observado que os quatro cenários descritos acima não são mutuamente exclusivos. Podem haver situações em que todos eles ocorrem ao mesmo tempo.

Após a atualização da fonte de dados f_P, sua próxima atualização é agendada, caso sua política de atualização seja *polling*. Nesse caso, um ponto importante deve ser observado. Quando a política de atualização do tipo *polling* foi descrita na Seção 4.1, foi mencionado o fato de que a monitoração na fonte de dados poderia ser feita seguindo parâmetros temporais fixos ou calculando-se a freqüência média com que a fonte de dados sofre alterações. Na primeira possibilidade, o agendamento da próxima atualização é feito com base nos parâmetros definidos quando as propriedades da fonte de dados foram definidas. Quando o agendamento da próxima atualização é feito pela freqüência média com que as alterações ocorrem na fonte, são levados em conta a média entre as alterações até o momento, a diferença de tempo entre a última e a atualização corrente da fonte de dados, e o número de atualizações da fonte, desde que a visão foi materializada. Logo, a fórmula dada pela expressão 4.3 abaixo determina qual será a nova freqüência média a ser considerada.

$$freq_{nova} = \frac{freq_{ant} + dif_{tempo}}{n_{atualiz} + 1} \tag{4.3}$$

onde:

$freq_{nova}$: novo valor assumido para a freqüência média com que as alterações ocorrem na fonte de dados;

$freq_{ant}$: valor da freqüência média anterior à atualização corrente da fonte de dados;
dif_{tempo}: diferença de tempo entre a última atualização e a atualização corrente da fonte de dados;
$n_{atualiz}$: número de atualizações da fonte de dados desde a materialização da visão.

Após o cálculo da nova freqüência média, o seu valor, assim como o da data da atualização corrente e o número de atualizações incrementado são armazenados em um arquivo XML, que armazena essas informações para cada fonte de dados cuja política de atualização é do tipo *polling*, para que o próximo cálculo possa ser feito corretamente. É intuitivo observar que, à medida que o número de atualizações da fonte de dados cresce, o valor da freqüência média converge para um valor mais próximo do real. Com isso é como se a ferramenta WebView estivesse "aprendendo" a freqüência com que a fonte de dados sofre alterações.

Atualizando uma Fonte de Dados Dependente

Como para uma fonte de dados primária, na atualização de uma fonte de dados dependente f_D, a primeira ação a ser executada pelo processador de visões é a coleta das páginas de f_D utilizando o agente de coleta $a(f_D)$, definido para a fonte. No entanto, uma diferença no processo de coleta deve aqui ser observada. No caso em que uma página deve ser coletada para cada objeto da fonte de dados mestre f_M, os atributos de junção entre as fontes f_D e f_M devem ser buscados em todos os objetos de f_M presentes na visão para que a coleta possa ser feita para cada um deles. Caso contrário, a coleta é feita normalmente pela simples execução do agente de coleta $a(f_D)$.

Coletadas as páginas, o extrator de dados $x(f_D)$ é então utilizado para a extração dos dados da fonte f_D. Os objetos extraídos nesse processo, assim como para uma fonte de dados primária, são utilizados para a atualização da visão.

Novamente, conideremos $O = \{o_1, o_2, \ldots, o_n\}$ o conjunto de objetos extraídos de f_D e $V = \{v_1, v_2, \ldots, v_m\}$ o conjunto de objetos já armazenados na visão cujo atributo *Primary Data Source* tem como valor o nome da fonte de dados primária f_P relacionada à fonte de dados f_D. Sejam ainda os conjuntos A_I dos atributos identificadores da fonte de dados f_D e A_J dos atributos definidos como atributos de junção entre a fonte dependente f_D e sua fonte de dados mestre f_M. Da mesma forma que para a atualização de uma fonte de dados primária, considerando-se os dois conjuntos distintos de objetos, os cenários abaixo são possíveis. Como conseqüência de cada um desses cenários, ações específicas são executadas para a correta atualização da visão.

(a) Existe um subconjunto $O' = \{o_1, o_2, \ldots, o_k\}$, $O' \subseteq O$ dos objetos extraídos, tal que $\nexists v_i \in V$, $1 \leq i \leq m$, tal que $v_i[A_I] = o_j[A_I]$, $\forall o_j \in O'$. Os objetos do conjunto O' são tais que foram extraídos da fonte f_D e ainda não estão presentes na visão, portanto, devem ser inseridos.

O conjunto de objetos O' é então considerado como uma fonte de dados f qualquer, de forma que ela deve ser integrada com as suas fontes de dados dependentes, da mesma forma que uma fonte de dados primária, resultando num repositório temporário f. Entretanto, um repositório temporário é criado contendo os objetos da fonte de dados mestre f_M e da fonte de dados mestre de f_M, sucessivamente, até que a fonte de dados primária correspondente seja alcançada. A operação de junção é executada sobre os dois repositórios temporários, resultando nos objetos que devem ser inseridos na visão.

(b) Existe um subconjunto $V' = \{v_1, v_2, \ldots, v_k\}$, $V' \subseteq V$ dos objetos da visão, tal que $\nexists o_i \in O$, $1 \leq i \leq n$, tal que $o_i[A_I] = v_j[A_I]$, $\forall v_j \in V'$. Os objetos do conjunto V' são tais que estavam presentes na visão e não estão mais presentes na fonte de dados, portanto, devem ser removidos.

O conjunto V' é constituído de objetos que não mais existem na fonte de dados f_D. Entretanto, esses objetos não são removidos da visão. Os atributos dos objetos de O devem ser convertidos para o valor nulo, assim como os atributos das fontes de dados dependentes de f_D. Os objetos de V' não são inteiramente removidos da visão pelo fato de que a operação de junção foi definida como uma junção à esquerda.

(c) Existe um subconjunto $O' = \{o_1, o_2, \ldots, o_k\}$, $O' \subseteq O$ dos objetos extraídos, tal que $\forall o_i \in O'$, $\exists v_j \in V$, tal que $v_j[A_I] = o_i[A_I]$. Os objetos do conjunto O' são tais que foram extraídos de f_D e já estavam presentes na visão. Seja A_J o conjunto dos atributos de junção entre a fonte f_D e a sua fonte de dados mestre f_M. Nesse caso, duas situações distintas podem ocorrer para cada objeto $o_i \in O'$:

(1) $o_i[A_J] \neq v_j[A_J]$. O objeto o_i na fonte de dados f_D não satisfaz mais as condições de junção entre as fontes de dados f_D e f_M. Nesse caso, se existe outro objeto $o_j \in O'$ que satisfaz as condições de junção entre f_D e f_M, então os atributos do objeto o_j substituem os atributos do objeto o_i em v_j. Com essa substituição, a junção entre f_D e alguma de suas fontes dependentes pode se tornar inconsistente, e deve ser refeita coletando e extraindo novos objetos dessa fonte. Caso

novos objetos não sejam encontrados, os seus atributos assumirão o valor nulo no objeto da visão v_j.

(2) $o_i[A_J] = v_j[A_J]$. O objeto o_i na fonte de dados f_D ainda satisfaz os critérios de junção entre as fontes de dados f_D e f_M. Entretanto, as condições de junção entre a fonte de dados f_D e alguma de suas dependentes pode não ser mais satisfeita. Dessa forma, como nas situações anteriores, a junção deve ser refeita, coletando e extraindo novos objetos das fontes dependentes de f_D, e, caso não existam, convertendo os atributos dos seus objetos para o valor nulo.

A próxima atualização da fonte de dados f_D é agendada após a atualização corrente, caso sua política de atualização seja *polling*. Aqui, a próxima atualização pode também ser determinada por parâmetros temporais fixos ou calculando-se a freqüência média com que a fonte de dados sofre alterações.

A atualização de visões de fontes de dados da Web, difere significativamente da atualização de visões em bancos de dados tradicionas [7, 8, 9, 25] pelo fato de que em bancos de dados tradicionais a visão e o banco de dados estão sob o mesmo domínio. Com isso, quando os dados básicos são alterados, sabe-se perfeitamente qual o impacto dessas alterações na definição da visão. Em visões de fontes de dados da Web, ao contrário, as visões e os dados básicos são independentes entre si, e alterações nos dados básicos não são conhecidas pelas visões. Por esse motivo, é que existe a necessidade de que quando os dados são coletados e extraídos de uma fonte de dados da Web seu conteúdo seja comparado com o conteúdo corrente da visão, para determinar quais foram as alterações. Nesse contexto, é que surgiu a necessidade da utilização dos atributos identificadores.

4.3 Visões Criadas Através da Ferramenta WebView

Esta seção é destinada à apresentação de alguns exemplos práticos de utilização da ferramenta WebView. São mostrados alguns exemplos de visões criadas com a ferramenta sobre *sites* populares encontrados na Web. O objetivo é mostrar como as características da ferramenta que foram discutidas ao longo deste capítulo podem ser abordadas na prática.

4.3.1 Programação de Cinema nas Cidades de Belo Horizonte e São Paulo

A primeira aplicação prática da ferramenta WebView apresentada é a criação e manutenção de uma visão que mantém a programação de cinema nas cidades de Belo Horizonte e São Paulo, além de informações técnicas relacionadas aos filmes. A informação é retirada de páginas do portal Terra[1].

A visão é composta de duas fontes de dados primárias: filmes em cartaz em Belo Horizonte e em São Paulo, uma fonte de dados de união, representando a união da programação nas duas cidades, além de duas fontes de dados dependentes, que provêm informação técnica sobre os filmes em cartaz. As Figuras 4.9 e 4.10 mostram exemplos do conteúdo dessas páginas.

Figura 4.9: Filmes em cartaz nas cidades de Belo Horizonte e São Paulo. Páginas encontradas no portal Terra.

O primeiro passo na definição da visão, é a definição do seu nome e diretório de trabalho, como mostra a Figura 4.11. O diretório de trabalho é o local onde a ferramenta irá procurar

[1]Terra - A Internet mais sua do que nunca. http://www.terra.com.br

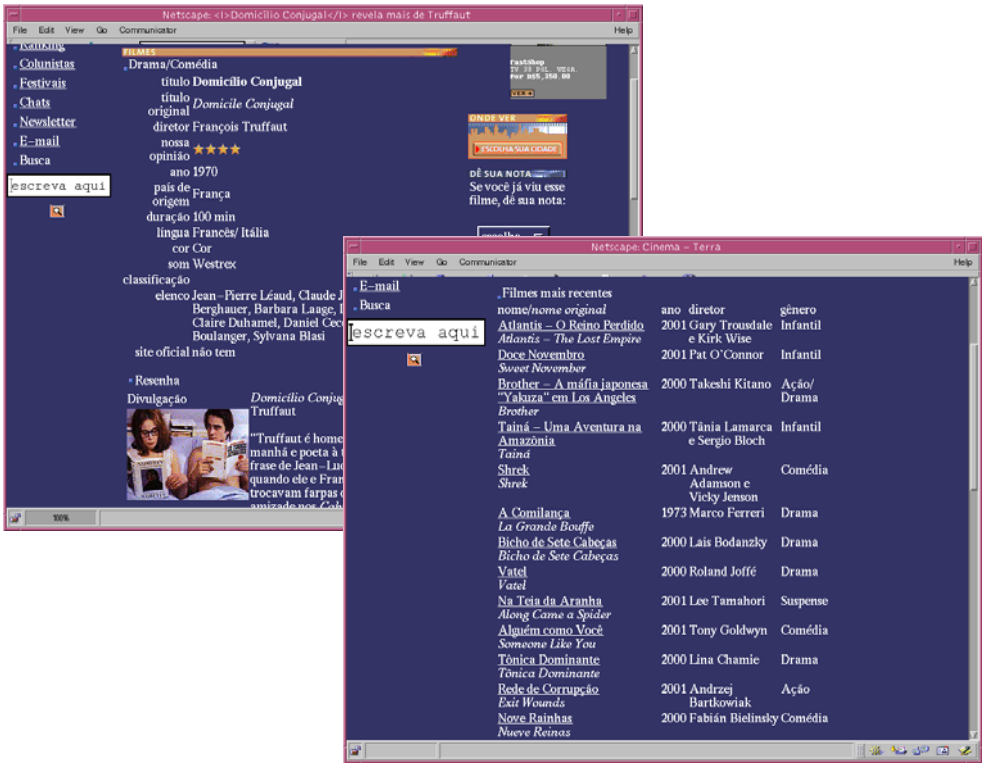


Figura 4.10: Informações técnicas sobre os filmes em cartaz. Páginas encontradas no portal Terra.

pelos agentes de coleta e extratores de dados de cada uma das fontes que constituem a visão. Pode-se observar ainda na Figura 4.11 a definição se a atualização da visão será incremental ou não.

Definidas essas propriedades da visão, o próximo passo é a inserção das fontes de dados que a constituem, assim como da definição de suas propriedades. As fontes de dados podem ser inseridas na visão através dos *menus* da interface.

Continuando a definição da visão, resta apenas definir as propriedades individuais de cada fonte de dados, estabelecendo o relacionamento entre elas. A Figura 4.3 mostra como o nome da fonte, o agente de coleta e o extrator de dados são definidos para fontes de dados primárias e dependentes.

Para cada fonte de dados primária, suas propriedades adicionais são definidas de acordo com a Figura 4.4. A fonte de dados de união, por sua vez, tem as propriedades definidas como mostra a Figura 4.7, basicamente estabelecendo quais são as fontes de dados que participam da união. Finalmente, são definidas as propriedades adicionais das fontes de dados dependentes, como mostra a Figura 4.5. Pode-se observar que o item Use join

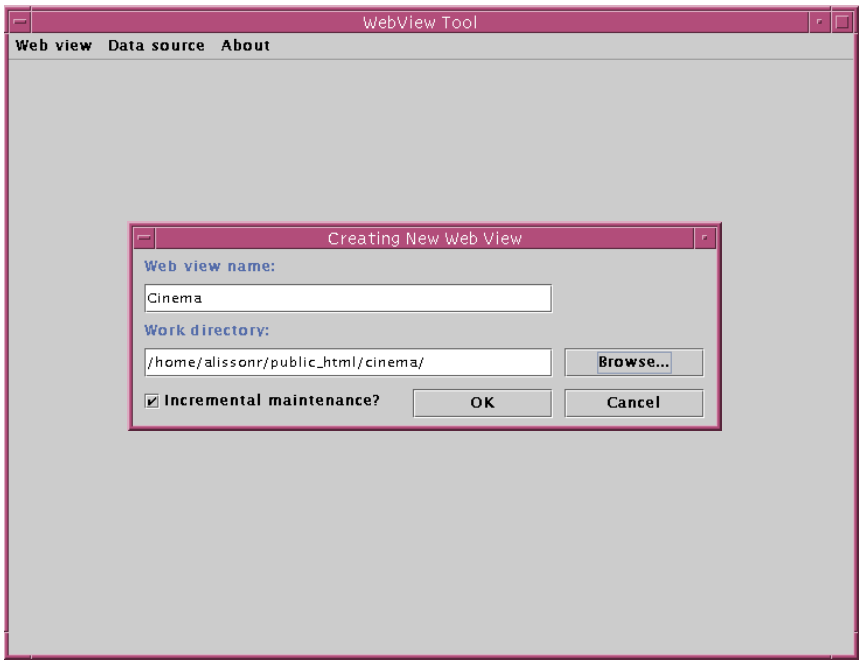


Figura 4.11: Definição do nome e diretório de trabalho de uma nova visão.

attributes for fetching? indica que uma página deve ser coletada para cada filme encontrado em cartaz.

Após a definição das propriedades de uma fonte de dados dependente, pode-se observar, na Figura 4.6, como são definidos os atributos de junção entre fontes de dados mestres e dependentes.

Ao final da definição de todas as fontes de dados, assim como de suas propriedades, o estado da janela principal da ferramenta WebView se apresenta como o da Figura 4.12, apresentando o grafo resultante da definição da visão.

Finalizando uma sessão de interação com a interface gráfica da ferramenta WebView, o plano de materialização pode ser gerado e a visão materializada. Essa visão pode ser facilmente estendida para acrescentar a programação cinematográfica de um número maior de cidades. Basta que haja uma página da Web para cada uma delas com a relação dos filmes em cartaz. A Figura 4.13 mostra o plano de materialização dessa visão, enquanto a Figura 4.14 mostra um trecho do arquivo XML segundo o formato DTORF [29] que armazena a visão.

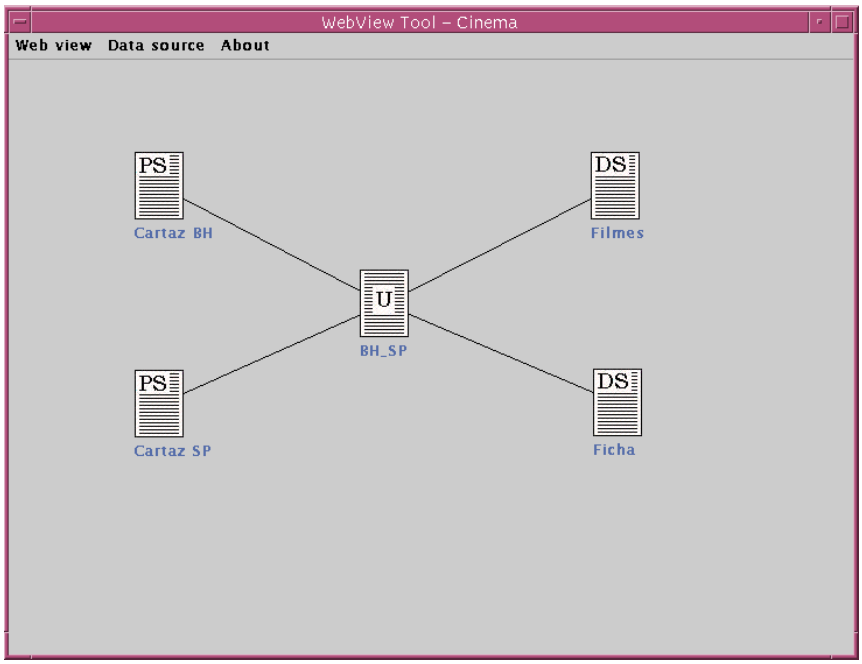


Figura 4.12: Estado final da janela principal da ferramenta WebView após a definição das fontes de dados e de suas propriedades para a visão relacionada à programação de cinema em Belo Horizonte e São Paulo.

4.3.2 Últimas Notícias do Futebol

A outra aplicação prática apresentada é a criação de uma visão que contém as últimas notícias relacionadas a futebol publicadas em uma página do portal Universo Online[2]. A Figura 4.15 apresenta exemplos do conteúdo dessas páginas. Essa é uma visão mais simples, que envolve apenas duas fontes de dados, uma delas primária, onde as manchetes das últimas notícias relacionadas a futebol são publicadas e a outra, dependente desta primeira, onde as notícias são efetivamente publicadas. O estado final da janela principal da ferramenta WebView para a definição desta visão pode ser visto na Figura 4.16.

A fonte de dados relativa às manchetes das últimas notícias é monitorada em intervalos de uma hora. Quando uma nova manchete é encontrada, a página relativa à notícia é coletada na outra fonte de dados, utilizando o título da notícia como atributo de junção. As notícias cujas manchetes não estejam mais na fonte são removidas da visão. A Figura 4.17 mostra o plano de materialização dessa visão, enquanto a Figura 4.18 mostra um trecho do arquivo XML segundo o formato DTORF [29] que armazena a visão.

[2]Universo Online - Esporte - Últimas do Futebol. http://www.uol.com.br/esporte/futebol/ultimas.htm

```
<WEBVIEW incremental="yes" name="Cinema">
    <PRIMARY identifiers="Titulo" name="Cartaz BH">
        <REFRESH policy="polling">
            <FREQUENCY calcavg="no" day="" dayofweek="Fri" hour="12" minute="00" month=""></FREQUENCY>
        </REFRESH>
        <FETCH agent="cartaz_bh.pl"></FETCH>
        <EXTRACT extractor="cartaz_bh.oep.xml"></EXTRACT>
    </PRIMARY>
    <PRIMARY identifiers="Titulo" name="Cartaz SP">
        <REFRESH policy="polling">
            <FREQUENCY calcavg="no" day="" dayofweek="Fri" hour="12" minute="00" month=""></FREQUENCY>
        </REFRESH>
        <FETCH agent="cartaz_sp.pl"></FETCH>
        <EXTRACT extractor="cartaz_sp.oep.xml"></EXTRACT>
    </PRIMARY>
    <UNION master="" name="BH_SP">
        <DATASOURCE name="cartaz bh"></DATASOURCE>
        <DATASOURCE name="cartaz sp"></DATASOURCE>
    </UNION>
    <DEPENDENT master="Bh_SP" name="Filmes">
        <REFRESH policy="triggering"></REFRESH>
        <FETCH agent="filmes.pl" usejoinattributes="no"></FETCH>
        <EXTRACT extractor="filmes.oep.xml"></EXTRACT>
        <JOIN>
            <ATTRIBUTE dependent="Titulo" master="Titulo"></ATTRIBUTE>
        </JOIN>
    </DEPENDENT>
    <DEPENDENT master="BH_SP" name="Ficha">
        <REFRESH policy="triggering"></REFRESH>
        <FETCH agent="ficha.pl" usejoinattributes="yes"></FETCH>
        <EXTRACT extractor="ficha.oep.xml"></EXTRACT>
        <JOIN>
            <ATTRIBUTE dependent="Titulo" master="Titulo"></ATTRIBUTE>
        </JOIN>
    </DEPENDENT>
</WEBVIEW>
```

Figura 4.13: Plano de materialização da visão relacionada à programação de cinema em Belo Horizonte e São Paulo.

4.3.3 Previsão do Tempo

Essa última visão apresentada, ainda mais simples, contém a previsão do tempo para as capitais brasileiras. A visão contém apenas uma única fonte de dados, composta por uma

```
<OBJECTS>
   <TUPLE type="BH_SP">
      <ATOM type="Primary Data Source">
         <VALUE>cartaz sp</VALUE>
      </ATOM>
      <ATOM type="Titulo">
         <VALUE fpos="50532" ipos="50522">A Partilha</VALUE>
      </ATOM>
      <ATOM type="Ano">
         <VALUE fpos="23302" ipos="23298">2001</VALUE>
      </ATOM>
      <ATOM type="Genero">
         <VALUE fpos="23015" ipos="23010">Drama</VALUE>
      </ATOM>
      <ATOM type="Titulo Original">
         <VALUE fpos="10247" ipos="10237">A Partilha</VALUE>
      </ATOM>
      <ATOM type="Pais de Origem">
         <VALUE fpos="11130" ipos="11124">Brasil</VALUE>
      </ATOM>
      <ATOM type="Duracao">
         <VALUE fpos="11292" ipos="11290">93</VALUE>
      </ATOM>
      <ATOM type="Lingua">
         <VALUE fpos="11459" ipos="11450">Portugues</VALUE>
      </ATOM>
      <ATOM type="Elenco">
         <VALUE fpos="12247" ipos="12094">Gloria Pires, Andrea Beltrao,...</VALUE>
      </ATOM>
   </TUPLE>
   ...
</OBJECTS>
```

Figura 4.14: Trecho do arquivo XML no formato DTORF [29] que armazena a visão relacionada à programação de cinema em Belo Horizonte e São Paulo.

página encontrada no *site* do Instituto Nacional de Meteorologia[3]. A Figura 4.19 mostra o conteúdo presente nesta fonte de dados.

A página deste *site* é atualizada diariamente com a previsão do tempo. Pode-se observar que esta é uma visão cuja atualização é do tipo não incremental. Não há objetos novos

[3]Meteorologia do INMET - Instituto Nacional de Meteorologia.
http://200.252.242.125/asp2/previsao.asp?tipo=2

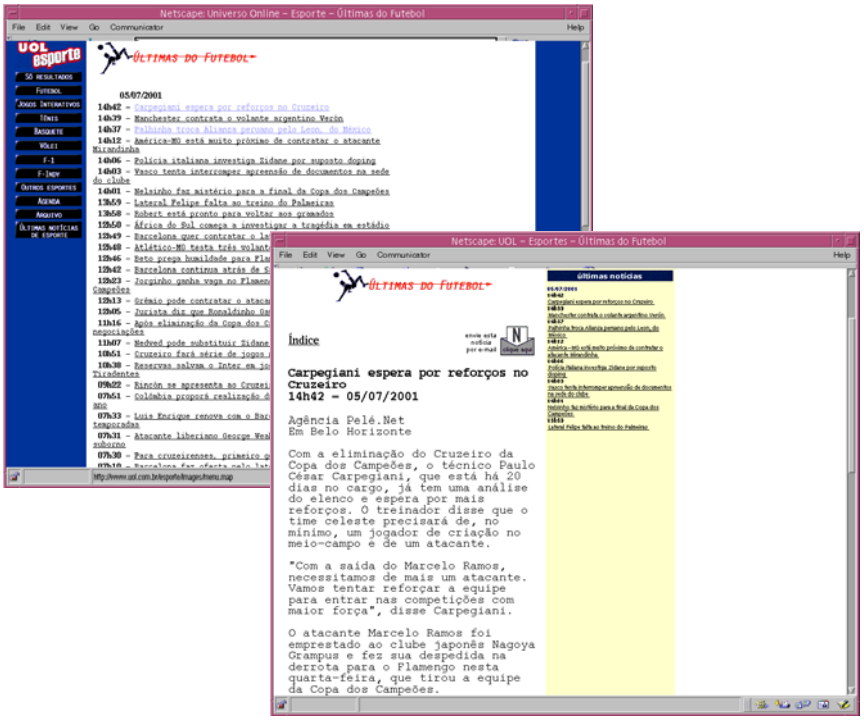


Figura 4.15: Manchetes e últimas notícias relacionadas a futebol. Páginas encontradas no portal Universo Online.

na fonte de dados que devam ser inseridos na visão e nem objetos na visão que devam ser removidos. Por outro lado, os objetos são sempre os mesmos, onde apenas alguns de seus atributos sofrem alterações. Dessa forma a visão é totalmente recomputada quando é atualizada. A Figura 4.20 mostra o plano de materialização dessa visão, enquanto a Figura 4.21 mostra um trecho do arquivo XML segundo o formato DTORF [29] que armazena a visão.

4.4 Aspectos de Implementação

Essa seção final é reservada à discussão de aspectos de implementação envolvidos no desenvolvimento da ferramenta WebView.

Conforme já foi amplamente discutido, a ferramenta WebView é composta por dois módulos: a sua interface gráfica e o processador de visões. Por características próprias de cada um dos módulos, uma linguagem de programação específica foi adotada na sua

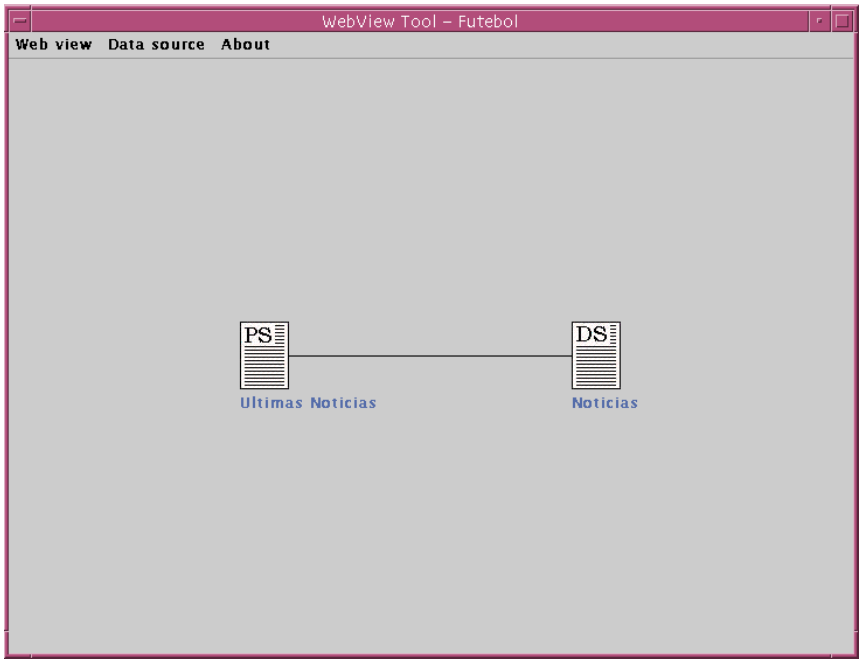


Figura 4.16: Estado final da janela principal da ferramenta WebView após a definição das fontes de dados e de suas propriedades para a visão relacionada às últimas notícias do futebol.

```
<WEBVIEW incremental="yes" name="Futebol">
    <PRIMARY identifiers="Titulo" name="Ultimas Noticias">
        <REFRESH policy="polling">
            <FREQUENCY calcavg="no" day="" dayofweek="" hour="" minute="00" month=""></FREQUENCY>
        </REFRESH>
        <FETCH agent="ultimas.pl"></FETCH>
        <EXTRACT extractor="ultimas.oep.xml"></EXTRACT>
    </PRIMARY>
    <DEPENDENT identifiers="Titulo" master="Ultimas Noticias" name="Noticias">
        <REFRESH policy="triggering"></REFRESH>
        <FETCH agent="noticias.pl" usejoinattributes="yes"></FETCH>
        <EXTRACT extractor="noticias.oep.xml"></EXTRACT>
        <JOIN>
            <ATTRIBUTE dependent="Titulo" master="Titulo"></ATTRIBUTE>
        </JOIN>
   </DEPENDENT>
</WEBVIEW>
```

Figura 4.17: Plano de materialização da visão relacionada às últimas notícias do futebol.

```
<OBJECTS>
   <TUPLE type="Ultimas Noticias">
      <ATOM type="Primary Data Source">
         <VALUE>Ultimas</VALUE>
      </ATOM>
      <ATOM type="Titulo">
         <VALUE fpos="3144" ipos="3102">Vasco consegue liminar para manter Edmundo</VALUE>
      </ATOM>
      <ATOM type="Hora">
         <VALUE fpos="4100" ipos="4095">15h39</VALUE>
      </ATOM>
      <ATOM type="Data">
         <VALUE fpos="4113" ipos="4103">13/07/2001</VALUE>
      </ATOM>
      <ATOM type="Agencia">
         <VALUE fpos="4142" ipos="4131">Lancepress!</VALUE>
      </ATOM>
      <ATOM type="Local">
         <VALUE fpos="4163" ipos="4146">No Rio de Janeiro</VALUE>
      </ATOM>
      <ATOM type="Conteudo">
         <VALUE fpos="4392" ipos="4171">De acordo com a assessoria de imprensa do Vasco...</VALUE>
      </ATOM>
   </TUPLE>
   ...
</OBJECTS>
```

Figura 4.18: Trecho do arquivo XML no formato DTORF [29] que armazena a visão relacionada às últimas notícias do futebol.

implementação.

A interface gráfica foi desenvolvida em linguagem Java [26], por permitir a manipulação de bibliotecas para a construção de aplicativos gráficos. Aplicações desenvolvidas em Java ainda apresentam a vantagem de serem independentes de arquitetura, o que permite a utilização multiplataforma da ferramenta.

O processador de visões, desenvolvido em linguagem Perl [43], tem como principal função a manipulação de arquivos texto, repositórios da extração dos dados das fontes da Web. Uma das principais vantagens da linguagem Perl é a grande facilidade para manipulação de estruturas de texto.

Além dos motivos expostos acima para a escolha das linguagens, ambas possuem biblio-

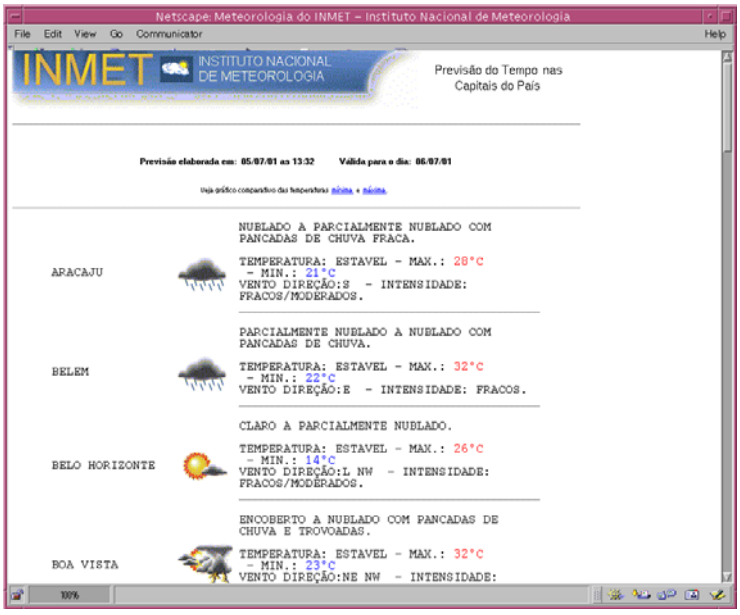


Figura 4.19: Previsão diária do tempo para as capitais brasileiras, feita pelo Instituto Nacional de Meteorologia.

```
<WEBVIEW incremental="no" name="Clima">
   <PRIMARY identifiers="Localidade" name="Tempo">
      <REFRESH policy="polling">
         <FREQUENCY calcavg="no" day="" dayofweek="" hour="14" minute="30" month=""></FREQUENCY>
      </REFRESH>
      <FETCH agent="tempo.pl"></FETCH>
      <EXTRACT extractor="tempo.oep.xml"></EXTRACT>
   </PRIMARY>
</WEBVIEW>
```

Figura 4.20: Plano de materialização da visão relacionada à previsão do tempo para as capitais brasileiras.

tecas para a manipulação de arquivos XML, como DOM (*Document Object Model*) [41]. Os dois módulos da ferramenta manipulam intensivamente arquivos XML. A interface gráfica realiza operações de leitura sobre os extratores de dados gerados pela ferramenta DEByE, que são armazenados em XML. A saída da interface gráfica é o plano de materialização da visão, que é também um arquivo XML. O processador de visões manipula os conjuntos de objetos gerados na extração das fontes de dados, armazenados em arquivos XML, realizando as operações de união e junção definidas sobre as fontes de dados.

Como as visões geradas pela ferramenta WebView são armazenadas em arquivos XML

```
<OBJECTS>
   <TUPLE type="Tempo">
      <ATOM type="Primary Data Source">
         <VALUE>tempo</VALUE>
      </ATOM>
      <ATOM type="Localidade">
         <VALUE fpos="1947" ipos="1940">ARACAJU</VALUE>
      </ATOM>
      <ATOM type="Situacao">
         <VALUE fpos="2167" ipos="2120">NUBLADO A PARCIALMENTE NUBLADO COM CHUVA FRACA.</VALUE>
      </ATOM>
      <ATOM type="Temperatura">
         <VALUE fpos="2198" ipos="2191">ESTAVEL</VALUE>
      </ATOM>
      <TUPLE type="Vento">
         <ATOM type="Direcao">
            <VALUE fpos="2354" ipos="2350">E SE</VALUE>
         </ATOM>
         <ATOM type="Intensidade">
            <VALUE fpos="2399" ipos="2382">FRACOS/MODERADOS.</VALUE>
         </ATOM>
      </TUPLE>
   </TUPLE>
   ...
</OBJECTS>
```

Figura 4.21: Trecho do arquivo XML no formato DTORF [29] que armazena a visão relacionada à previsão do tempo para as capitais brasileiras.

no formato TOR (*Textual Object Repository*), atualizações na visão são atualizações nesse arquivo XML. Quando uma visão deve ser atualizada todo o seu conteúdo é carregado em memória utilizando bibliotecas que implementam DOM(*Document Object Model*). Essas bibliotecas armazenam o conteúdo do documento XML em uma estrutura hierárquica. A partir dessa representação hierárquica do conteúdo do documento XML, que na verdade é o conteúdo da visão, a visão pode ser atualizada. Os objetos extraídos das fontes de dados que correspondem aos dados encontrados nas fontes que ainda não estão na visão são inseridos na estrutura hierárquica. Os objetos que não estão mais presentes nas fontes de dados, e portanto devem ser removidos da visão, são removidos da estrutura hierárquica. Após as alterações, a estrutura hierárquica atualizada em memória é novamente armazenada em um documento XML no formato TOR, que é o novo conteúdo atualizado da visão.

Capítulo 5

Conclusões

Neste capítulo é feita uma revisão do trabalho desenvolvido, analisando os objetivos propostos inicialmente e como eles foram atingidos. Além dessa revisão, são discutidas as principais contribuições do trabalho e possíveis direções para trabalhos futuros.

5.1 Revisão do Trabalho

O objetivo do trabalho proposto foi o desenvolvimento de um ambiente a partir de onde usuários pudessem criar visões de fontes de dados da Web. O ambiente desenvolvido teria ainda a responsabilidade de manter as visões atualizadas, consistentes com as fontes de dados originais. A principal motivação para a criação desse ambiente foi discutida na Seção 1.2. Existe uma grande variedade de aplicações que utilizam dados provenientes da Web. Devido à sua natureza altamente dinâmica, a atualização desses dados de forma automática pelo ambiente desenvolvido apresenta grande aplicabilidade nesse contexto. Além disso, por se tratar de um ambiente distribuído de informação, a integração das fontes de dados da Web também é importante.

No Capítulo 2, foi feita uma revisão do conceito de visões e de suas aplicações na tecnologia de bancos de dados. Foram discutidos vários aspectos relacionados à implementação de visões por um SGBD. Em seguida, essa discussão foi estendida para o contexto de visões de fontes de dados da Web. O que se pôde perceber é que, embora relativamente simples, o conceito de visão tem grande aplicabilidade e utilidade na tecnologia de bancos de dados, e a sua adoção ao contexto da Web se mostrou uma abordagem extremamente promissora.

O Capítulo 3 apresentou a arquitetura do ambiente proposto para a criação e ma-

nutenção de visões de fontes de dados da Web. Foram apresentadas as ferramentas que compõem este ambiente. A ferramenta ASByE (*Agent Specification By Example*) [22, 23] é utilizada para a criação de agentes para a coleta automática de páginas da Web. A ferramenta DEByE (*Data Extraction By Example*) [29, 39] tem por objetivo a criação de extratores de dados que, quando executados sobre as páginas coletadas, extraem os dados semi-estruturados presentes nestas páginas. A ferramenta WebView [3, 4], a idéia central deste trabalho, permite criar visões de fontes de dados da Web. A ferramenta ainda executa as ações necessárias para que essas visões sejam mantidas consistentes com as fontes de dados originais, utilizando agentes de coleta gerados pela ferramenta ASByE e extratores de dados gerados pela ferramenta DEByE.

Finalmente, o Capítulo 4 descreveu em um nível maior de detalhes a ferramenta WebView. Foram descritos os dois módulos componentes da ferramenta: a sua interface gráfica e o processador de visões. A interface gráfica da ferramenta, baseada numa metáfora de grafos, permite que visões de fontes de dados da Web sejam modeladas e criadas. A saída gerada por essa interface é o plano de materialização da visão, um arquivo XML que codifica a informação necessária sobre a visão e as fontes de dados que a constituem. Esse plano de materialização é a entrada do processador de visões. Através da execução desse plano, o processador de visões executa os agentes de coleta e extratores de dados de cada uma das fontes com o objetivo de materializar e manter a visão consistente. Nesse capítulo, foram ainda apresentados exemplos de visões criadas através da ferramenta, além de aspectos envolvidos na sua implementação.

Fazendo uma análise global de todas as visões apresentadas na Seção 4.3, pode-se observar que, em geral, visões de fontes de dados da Web envolvem um pequeno número de fontes de dados. Mais uma vez isso acontece em conseqüência do fato de que a Web não é um banco de dados, embora se esteja procurando tratá-la como tal. As fontes de dados da Web apresentam contextos muito bem definidos e, na grande maioria das vezes, seu conteúdo de informação é satisfatório. Isso nos leva a muitas vezes criar visões com um pequeno número de fontes de dados, ou até mesmo com informação redundante entre elas. Esse cenário é totalmente o oposto ao ambiente de bancos de dados tradicionais. Nesse ambiente, em conseqüência da utilização de conceitos como a normalização dos dados [15], por exemplo, os dados são distribuídos entre as fontes procurando eliminar redundâncias. Assim, quando uma visão deve ser criada para abranger um contexto bem específico de informação, um grande número de fontes de dados deve estar envolvido.

De acordo com [12], uma das grandes dificuldades no desenvolvimento de um sistema

de informação baseado na Web é a localização de fontes de dados úteis para o contexto em questão, à medida que se deseja fornecer uma maior cobertura para o sistema.

5.2 Principais Contribuições

Neste trabalho foi desenvolvido um ambiente [3, 4] para criação e manutenção de visões de fontes de dados da Web. Através desse ambiente, o conceito de visão, amplamente utilizado em bancos de dados tradicionais [15], pode ser adotado no contexto de fontes de dados da Web. Para isso, foram desenvolvidas técnicas que permitem que dados provenientes da Web sejam materializados localmente e automaticamente atualizados em relação às fontes de dados originais. Além disso, dados provenientes de fontes de dados distintas podem ser integrados em uma única visão. Com essa integração, a formulação de consultas sobre diversas fontes de dados da Web, mas relacionadas entre si, se torna muito mais simples, em virtude da integração dos dados dessas fontes em uma única visão. O resultado dessa integração é na verdade um repositório TOR (*Textual Object Repository*), um arquivo XML que implementa o modelo de dados DEByE-OM [29]. Com isso, esse resultado pode ser consultado utilizando alguma linguagem ou interface de consulta para XML [17, 32, 33], armazenado em bancos de dados relacionais [34, 35] ou republicado na Web utilizando XSL [41].

A abordagem foi desenvolvida utilizando o conceito de visão localmente materializada. Algumas abordagens similares, entretanto, utilizam a abordagem virtual, onde os dados são coletados, extraídos e integrados durante o processamento da consulta [21, 27]. Acredita-se que a abordagem materializada é mais vantajosa, em virtude dos dados estarem materializados localmente. Com isso, as técnicas de manutenção de visões desenvolvidas permitem que alterações nas fontes de dados sejam refletidas nas visões. Pode-se ainda obter ganho em eficiência utilizando a manutenção incremental de uma visão. Somente as partes alteradas das fontes de dados são propagadas para a visão, o que torna desnecessária a sua total rematerialização.

A abordagem materializada desenvolvida apresenta as seguintes vantagens: (1) diminui o tempo de resposta a consultas, em virtude da localização dos dados; (2) evita a navegação exaustiva, em virtude da integração das fontes de dados; (3) permite que os dados estejam acessíveis mesmo quando as fontes de dados originais não estejam; e (4) diminui o tráfego na rede, por eliminar a necessidade do acesso direto às fontes de dados.

O ambiente para a criação e manutenção de visões desenvolvido é totalmente baseado

na utilização de ferramentas de alto nível. Dessa forma, a escrita manual de código não é necessária, o que o torna mais conveniente para a utilização por usuários menos especializados e para o rápido desenvolvimento de sistemas de informação baseados na Web, quando comparado com abordagens relacionadas.

As fontes de dados da Web, sobre as quais as visões são definidas, foram categorizadas de acordo com o seu comportamento em uma visão. Essa categorização constitui parte essencial do trabalho. Com isso, fontes de dados distintas podem ser relacionadas e os dados provenientes dessas fontes podem ser integrados. Essa categorização das fontes de dados é discutida em detalhes no Capítulo 4.

Finalmente, o trabalho desenvolvido foi realizado tendo as ferramentas ASByE e DEByE como base. Com isso, à medida que o trabalho foi se concretizando, diversas contribuições para a melhoria dessas ferramentas puderam ser detectadas. Entre essas contribuições estão o auxílio na correção de erros de implementação e melhorias nas interfaces das ferramentas. Ainda relacionado à ferramenta ASByE, diversos tipos de agentes de coleta de páginas ainda não oferecidos pela ferramenta foram detectados e adicionados à ferramenta. Além disso, houve uma maior integração das ferramentas ASByE e DEByE, que até então eram independentes.

5.3 Trabalhos Futuros

O principal trabalho futuro a ser seguido é a melhoria do critério de identidade entre objetos em fontes de dados distintas. Sabe-se que considerar dois objetos de fontes de dados distintas como sendo a mesma entidade do mundo real apenas se eles tiverem seus atributos de junção exatamente iguais é uma abordagem restrita, quando se trata de fontes de dados heterogêneas, como é o caso da Web. Esse talvez seja um dos grandes problemas na integração de fontes de dados heterogêneas. Algumas possíveis soluções aparecem na literatura e podem ser adotadas, como, por exemplo, a adoção da similaridade textual entre os objetos [11].

Por se tratar de um ambiente altamente dinâmico e aberto, as fontes de dados da Web podem sofrer alterações imprevisíveis. Essas alterações podem incluir, além de alterações em seu conteúdo, que são as alterações tratadas pela ferramenta WebView, alterações estruturais nas fontes de dados. Portanto, é desejável que se tenha mecanismos para a detecção da necessidade da manutenção adequada nos agentes de coleta e extratores de uma fonte de dados.

Freqüentemente, os mesmos dados de uma fonte dependente podem estar relacionados a dados diferentes na fonte de dados mestre, ou ainda a dados provenientes de fontes de dados mestres distintas. Isso leva a armazenar dados redundantes numa visão, quando a operação de junção entre as fontes é realizada. Para evitar essa redundância, pode-se utilizar os atributos ID e IDREF da linguagem XML [41] nos dados armazenados na visão. Com a utilização desses atributos, é possível fazer referência entre objetos. Dessa forma, pode-se eliminar esse problema, diminuindo a redundância dos dados armazenados.

Outro ponto importante que pode ser abordado como trabalho futuro, é a questão da análise do desempenho e escalabilidade da ferramenta WebView. Assim como em bancos de dados distribuídos [15], a ordem em que as junções entre fontes de dados é executada pode ter impacto no desempenho obtido. Com isso, determinar a melhor ordem em que as fontes de dados devem ser integradas é um fator importante a ser desenvolvido.

Finalmente, pode-se estender o ambiente desenvolvido de forma a capacitá-lo à manipulação de imagens, freqüentemente encontradas nas fontes de dados da Web, como é feito nos sistemas ARIADNE [27] e ARANEUS [36], por exemplo.

Bibliografia

[1] Abiteboul, S. On Views and XML. In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Philadelphia, Pennsylvania, 1999), ACM Press, pp. 1–9.

[2] Abiteboul, S., Buneman, P., and Suciu, D. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, Los Altos, California, 1999.

[3] Arantes, A. R., Laender, A. H. F., Golgher, P. B., and da Silva, A. S. An Environment for Building and Maintaining Web Views. In *Proceedings of the International Workshop on Information Integration on the Web (WIIW 2001)* (Itaipava, Rio de Janeiro, 2001), pp. 172–178.

[4] Arantes, A. R., Laender, A. H. F., Golgher, P. B., and da Silva, A. S. Managing Web Data Through Views. In *Electronic Commerce and Web Technologies, Second International Conference, EC-Web 2001* (Berlin, Germany, 2001), S. K. Mandria and G. Pernul, Eds., Springer, pp. 154–165.

[5] Atzeni, P., Masci, A., Mecca, G., Merialdo, P., and Tabet, E. ULIXES: Building Relational Views over the Web. In *Proceedings of the Thirteenth International Conference on Data Engineering* (Birmingham, U.K., 1997), A. Gray and P.-Å. Larson, Eds., IEEE Computer Society, p. 576.

[6] Bhowmick, S. S., Madria, S. K., Ng, W. K., and Lim, E. P. Web Warehousing: Design and Issues. In *Advances in Database Technologies, ER'98 Workshops on Data Warehousing and Data Mining* (Berlin, Germany, 1998), D. Lee, E. Lim, M. Mohania, and Y. Masunaga, Eds., Springer, pp. 93–104.

[7] Blakeley, J. A., Coburn, N., and Larson, P.-Å. Updating Derived Relations: Detecting Irrelevant and Autonomously Computable Updates. *ACM Transactions On Database Systems 14*, 3 (1989), 369–400.

[8] Blakeley, J. A., Larson, P. A., and Tompa, F. W. Efficiently Updating Materialized Views. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data* (Washington, D.C., 1986), C. Zaniolo, Ed., pp. 61–71.

[9] Ceri, S., and Widom, J. Deriving Production Rules for Incremental View Maintenance. In *Proceedings of the Seventeenth International Conference on Very Large Data Bases* (Barcelona, Catalonia, Spain, 1991), G. M. Lohman, A. Sernadas, and R. Camps, Eds., Morgan Kaufmann, pp. 577–589.

[10] Chaves, K. G. Realimentação de Exemplos na Extração de Dados Semi-Estruturados. Dissertação de Mestrado, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Julho 2000.

[11] Cohen, W. W. Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Seattle, Washington, 1998), pp. 201–212.

[12] Cohen, W. W. Some Practical Observations on Integration of Web Information. In *Proceedings of the ACM SIGMOD Workshop on The Web and Databases (WebDB'99)* (Philadelphia, Pennsylvania, 1999), S. Cluet and T. Milo, Eds., pp. 55–60.

[13] Date, C. J. *An Introduction to Data Base Systems*, 7 ed., vol. 1. Addison-Wesley, Reading, Massachusetts, 2000.

[14] Dayal, U., and Bernstein, P. A. On the Updatability of Relational Views. In *Proceedings of the Fourth International Conference on Very Large Data Bases* (West Berlin, Germany, 1978), S. B. Yao, Ed., IEEE Computer Society, pp. 368–377.

[15] Elmasri, R., and Navathe, S. B. *Fundamentals of Database Systems*, 3 ed. Addison Wesley, Menlo Park, California, 2000.

[16] Embley, D. W., Campbell, D. M., Jiang, Y. S., Liddle, S. W., Ng, Y.-K., Quass, D., and Smith, R. D. Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages. *Data Knowledge Engineering 31*, 3 (1999), 227–251.

[17] Evangelista-Filha, I. M. R. Uma Interface Gráfica para Consulta a Dados Semi-Estruturados através de Exemplos. Dissertação de Mestrado, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Maio 2001.

[18] Evangelista-Filha, I. M. R., Laender, A. H. F., and da Silva, A. S. Querying Semistructured Data By Example: The QSByE Interface. In *Proceedings of the International Workshop on Information Integration on the Web (WIIW 2001)* (Itaipava, Rio de Janeiro, 2001), pp. 156 –163.

[19] Franklin, M. J., and Zdonik, S. B. "Data In Your Face": Push Technology in Perspective. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Seattle, Washington, 1998), L. M. Haas and A. Tiwary, Eds., ACM Press, pp. 516–519.

[20] Furtado, A. L., and Casanova, M. A. Updating Relational Views. In *Query Processing in Database Systems*, W. Kim, D. S. Reiner, and D. S. Batory, Ed. Springer, Berlin, 1985, pp. 127–142.

[21] García-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., and Widom, J. Integrating and Accessing Heterogeneous Information Sources in TSIMMIS. In *Proceedings of the AAAI Spring Symposium on Information Gathering* (Stanford, California, 1995), pp. 61–64.

[22] Golgher, P. B., Laender, A. H. F., da Silva, A. S., and Ribeiro-Neto, B. An Example-Based Enviroment for Wrapper Generation. In *Conceptual Modeling for E-Business and the Web, ER 2000 Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling* (Berlin, Germany, 2000), S. Liddle, H. Mayr, and B. Thalheim, Eds., Springer, pp. 94–101.

[23] Golgher, P. B., Laender, A. H. F., da Silva, A. S., and Ribeiro-Neto, B. ASByE: Uma Ferramenta Baseada em Exemplos para Especificação de Agentes para Coleta de Documentos Web. In *Anais do XV Simpósio Brasileiro de Bancos de Dados* (João Pessoa, Paraíba, 2000), pp. 217–231.

[24] Gupta, A., Harinarayan, V., and Rajaraman, A. Virtual Database Technology. In *Proceedings of the Fourteenth International Conference on Data Engineering* (Orlando, Florida, 1998), IEEE Computer Society, pp. 297–301.

[25] Gupta, A., Mumick, I. S., and Subrahmanian, V. S. Maintaining Views Incrementally. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data* (Washington, D.C., 1993), P. Buneman and S. Jajodia, Eds., pp. 157–166.

[26] java.sun.com - The Source for Java(TM) Technology. http://www.java.sun.com/.

[27] Knoblock, C., and Minton, S. The Ariadne Approach to Web-based Information Integration. *IEEE Intelligent Systems 13*, 5 (1998), 17–20.

[28] Lacroix, Z. Object Views Through Search Views of Web Datasources. In *Proceedings of the Eighteenth International Conference on Conceptual Modeling* (Paris, France, 1999), J. Akoka, M. Bouzeghoub, I.Comyn-Wattiau, and E. Métais, Eds., vol. 1728 of *Lecture Notes in Computer Science*, Springer, pp. 176–187.

[29] Laender, A. H. F., Ribeiro-Neto, B., da Silva, A. S., and Silva, E. S. Representing Web Data as Complex Objects. In *Electronic Commerce and Web Technologies, First International Conference EC-Web 2000*, K. Bauknecht, S. K. Madria, and G. Pernul, Eds. Springer, Berlin, Germany, 2000, pp. 216–228.

[30] Langerak, R. View Updates in Relational Databases with an Independent Scheme. *ACM Transactions On Databases Systems 15*, 1 (1990), 40–66.

[31] Lawrence, S., and Giles, C. Searching the World Wide Web. *Science 280*, 4 (1998), 98–100.

[32] Louly, K. G. Uma Interface Gráfica para Consulta a Fontes de Dados XML. Dissertação de Mestrado, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Março 2001.

[33] Louly, K. G., Laender, A. H. F., da Silva, A. S., and Coelho, T. A. S. Uma Interface Gráfica para Consulta a Fontes de Dados XML. In *Anais do XV Simpósio Brasileiro de Bancos de Dados* (João Pessoa, Paraíba, 2000).

[34] Magalhães, K. V. Uma Abordagem para Armazenamento de Dados Semi-Estruturados em Bancos de Dados Relacionais. Dissertação de Mestrado, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Março 2001.

[35] Magalhães, K. V., Laender, A. H. F., and da Silva, A. S. Storing Semistructured Data in Relational Databases. In *Proceedings of the String Processing and Information Retrieval Symposium (SPIRE 2001)* (Laguna de San Rafael, Chile, 2001).

[36] Mecca, G., Atzeni, P., Masci, A., Merialdo, P., and Sindoni, G. The ARANEUS Web-Base Management System. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Seattle, Washington, 1998), L. M. Haas and A. Tiwary, Eds., ACM Press, pp. 544–546.

[37] Melton, J., and Simon, A. R. *Understanding the New SQL: A Complete Guide.* Morgan Kaufmann, San Mateo, 1993.

[38] P. Buneman. Semistructured Data. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Tucson, Arizona, 1997), ACM Press, pp. 117–121.

[39] Ribeiro-Neto, B., Laender, A. H. F., and da Silva, A. S. Extracting Semi-Structured Data Through Examples. In *Proceedings of the Eighth ACM International Conference on Information and Knowledge Management - CIKM'99* (Kansas City, Missouri, 1999), pp. 94–101.

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

[41] Spencer, P. *XML design and implementation: programming with XML, ASP, and IE5.* Programmer to programmer. Wrox Press, Chicago, IL, 1999.

[42] Stonebraker, M. Implementation of Integrity Constraints and Views by Query Modification. In *Proceedings of the 1975 ACM SIGMOD International Conference on Management of Data* (San Jose, California, 1975), W. F. King, Ed., ACM, pp. 65–78.

[43] Wall, L., and Schwartz, R. L. *Programming Perl.* O'Reilly Associates, Inc., Sebastopol, CA, 1990.

[44] Wolfson, O., Sistla, P., Dao, S., Narayanan, K., and Raj, R. View Maintenance in Mobile Computing. *SIGMOD Record 24*, 4 (September 1995), 22–27.

[45] Zloof, M. M. Query-by-Example: A Data Base Language. *IBM Systems Journal 16*, 4 (1977), 324–343.