Tecnologia de Mediação para Extração e Transformação de Dados em Ambientes de Data Warehouse

Letícia Maria Gonçalves Furtado
Ana Maria de Carvalho Moura
Instituto Militar de Engenharia – IME/RJ
Departamento de Engenharia de Sistemas
Rio de Janeiro, Brasil
e-mail: leticiag.furtado@bol.com.br, anamoura@ime.eb.br

Abstract

Governmental and environmental agencies have recognized that Data Warehouses (DWs) oriented to environmental domain can cause a deep impact in the ability to preserve the environment. However, extracting and transforming environmental data from external sources in order to load them into the data organization area of a DW are much more complex when compared with conventional DWs. In environmental DWs, data are based on autonomous, heterogeneous external data sources distributed over the Web. In order to improve data extraction and transformation, this work proposes the conjointly use of a middleware system and DW technologies. Hence a middleware system locates and extracts Web data sources with their respective metadata, and applies on them a special a transformation service developed in the scope of this work as *program wrappers*, before loading them into the ODS (Operational data Store) of a DW. Furthermore, this transformation process is described using CWM (Common Warehouse Metamodel), a metadata standard to provide interoperability between DW development tools.

Key words: Data Transformation, Data Warehouse, Metadata, CWM, Mediators.

Resumo

Agências e organizações ambientais reconhecem que Data Warehouses (DWs) orientados a assuntos ambientais podem causar profundo impacto na habilidade de proteger o meio ambiente. No entanto, a extração e transformação de dados ambientais das fontes externas para carga na área de organização de dados do DW são muito mais complexos quando comparados aos DWs convencionais. Nos DWs ambientais, os dados são baseados em fontes de dados externas, autônomas e heterogêneas distribuídas na *Web*. Para melhorar a extração e transformação de dados, este trabalho propõe o uso conjunto do sistema de *middleware* e da tecnologia de DW. O sistema de *middleware* localiza e extrai dados de fontes *Web* com seus respectivos metadados, e aplica sobre eles um serviço de transformação desenvolvido no escopo desse trabalho como *wrapper* de programa, depois os carrega no Operation Data Store (ODS) do DW. Esse processo de transformação é descrito usando CWM, um padrão de metadados para prover interoperabilidade entre ferramentas de desenvolvimento de DW.

Palavras Chaves: Transformação de Dados, Data Warehouse, Metadados, CWM, Mediadores.

1. Introdução

A literatura aponta várias definições para um Data Warehouse (DW) [9], [11], [18], porém um DW pode de fato ser considerado como um banco de dados corporativo que contém dados extraídos do ambiente de produção da empresa, pré-selecionados, depurados e otimizados para o processamento de consulta e não para o processamento de transações. Em geral, um DW requer a consolidação de outros recursos de dados além daqueles armazenados em banco de dados relacionais, incluindo informações provenientes de planilhas eletrônicas, documentos textuais, fontes *Web*, etc. DWs visam proporcionar ao usuário uma visão analítica dos dados (normalmente dados históricos) integrados, oferecendo os fundamentos e recursos necessários a um Sistema de Apoio à Decisão (SAD).
A extração, limpeza, transformação e migração dos dados dos sistemas existentes na empresa para o DW constituem tarefas críticas para o seu funcionamento efetivo e eficiente. Porém, o grande desafio atual na construção de DWs consiste da complexidade nos processos de extração e integração de dados, principalmente quando estes são provenientes de fontes *Web* [6]. Esta tarefa torna-se ainda mais complexa quando se considera, por exemplo, o desenvolvimento de Sistemas de Apoio a Decisão ambientais (SADs). Estes são construídos a partir de DW ambientais, constituídos de dados geo-referenciáveis, de meio ambiente (a exemplo de temperatura, níveis pluviométricos, índices de chuvas, etc.), que envolvem: um grande volume de dados altamente distribuídos a ser processado; dados armazenados em uma grande variedade de formatos, sem padrões pré-definidos, necessitando ainda serem integrados e compartilhados entre sistemas de plataformas distintas, etc.
Várias são as tecnologias computacionais que podem ser utilizadas para a integração de fontes heterogêneas [17] [19] [4] [3]. Da abordagem que preconiza a integração sem materialização, também denominados sistemas virtuais, fazem parte os Sistemas Interoperáveis (Mediadores e *Middlewares*), que provêem uma interface uniforme para acesso aos dados [7]. Os mediadores e *middlewares* [8] [15] [12] utilizam as funcionalidades de componentes de software, genericamente chamados de tradutores, para acessar e traduzir fontes de dados heterogêneas, distribuídas e autônomas, tornando possível a criação de uma visão integrada dessas fontes. Dentro desse grupo de tecnologia está inserido o sistema de *middleware Le Select* [13], cuja principal proposta é publicar e disponibilizar fontes de dados e seus respectivos metadados para consulta por parte de usuários e/ou aplicações clientes, segundo uma visão relacional.
Esse trabalho propõe o uso conjunto de um sistema de mediação e da tecnologia de DW no processo de transformação de dados extraídos pelo *Le Select*. Dessa forma, o sistema localiza a(s) fonte(s) *Web* desejada(s), extrai os dados e metadados correspondentes sob forma de tabelas relacionais, e aplica as transformações necessárias a esses dados antes de carregá-los no ODS (*Operational Data Store*) de um DW.
O restante deste artigo está descrito da seguinte forma: a seção 2 faz uma descrição sucinta sobre transformações de dados em ambiente de DW. A seção 3 apresenta as principais funcionalidades do *middleware Le Select*, mostrando como as operações de transformação são executadas nesse ambiente. A seção 4 destaca a importância dos metadados no DW e descreve o padrão de metadados CWM, proposto pela W3C como padrão para a catalogação de metadados nos diversos processos envolvendo um DW. A seção 5 descreve o desenvolvimento dos processos de transformação implementados como *wrappers* de programas, bem como os descreve utilizando o padrão CWM. Finalmente, a seção 6 apresenta as conclusões do artigo com as suas principais contribuições e sugestões para trabalhos futuros.

2. Transformação de dados

Os processos ETL (extração, transformação e carga) são considerados como uma das fases mais complexas do ciclo de vida de um DW, que se justificam por dois fatores: a variedade dos sistemas fontes e de banco de dados que devem ser acessados; e as transformações e os critérios de qualidade que devem ser abordados para preparar os dados para o DW [10].
No contexto deste trabalho a fase de extração de dados é realizada pelo *Le Select*, após a qual aplicam-se transformações sobre estes dados. Neste artigo são tratadas as transformações mais comumente utilizadas em um DW, tais como [10]:

- **conversão de formatos:** esse tipo de transformação é de grande importância em DWs ambientais, quando a mudança entre diferentes unidades de medida é conveniente (temperatura de graus Celsius para Kelvin, por exemplo). Nesse caso é importante a implementação de operações básicas tais como soma, subtração, divisão e multiplicação que atuam entre uma coluna e uma constante ou entre duas colunas;
- **aplicação de faixas de intervalos:** essa operação é relevante em DWs quando um intervalo pode ser transformado em um valor único. Como exemplo de aplicação pode-se citar a transformação denominada Faixa Etária, que a partir de uma coluna de idades, por exemplo, transforma-as em uma *string* de caracteres segundo uma classificação prévia. Por exemplo, idades menores ou iguais a 12 são substituídas pela palavra "criança"; idades maiores que 12 e menores ou iguais a 18 por "adolescente", etc.
- **transformações de *string* para *string*:** esse tipo de transformação é muito utilizada na etapa de filtragem e padronização de termos em DW. Considere por exemplo a coluna sexo, onde é necessário mudar a representação final dos elementos dessa coluna, substituindo 0 ou M por "masculino" e 1 ou F por "feminino".

- **transformações de *string* para double:** nesse caso, onde existir um valor de coluna definido como null pode-se transformá-lo em um double com valor 0, de modo a permitir operações aritméticas sobre esta coluna.

Uma vez no formato apropriado, os dados podem ser carregados para as tabelas de destino no DW.
A próxima seção fará um *overview* sobre o *middleware Le Select*, mostrando como estas transformações são executadas neste ambiente.

3. O Sistema de Middleware Le Select

Le Select é um *framework* para acessar dados de natureza heterogênea e para invocar programas de processamento de dados em ambientes de *Internet/Intranet* [13]. Suas principais características são: acesso uniforme a dados heterogêneos, com suporte a vários tipos de fontes de dados; disponibilização e acesso a esses dados através de consultas *ad hoc* e/ou programas; e ainda a invocação de programas a conjuntos arbitrários de dados.
Como muitos outros sistemas de *midleware*, o *Le Select* oferece meios para integrar fontes de dados heterogêneas, facilitando ao máximo a tarefa do usuário dos dados. Porém, diferentemente de outros sistemas de mediação, o *Le Select* é completamente distribuído. Não existe repositório de publicação de dados centralizado e nem tão pouco um esquema global integrado. Porém, muitos servidores podem existir, cooperando para proverem acesso a dados e programas.
Publicar dados consiste na operação de disponibilizá-los no servidor *Le Select*. Assim, dados e programas são publicados em *sites*, desde que exista um servidor *Le Select* rodando. Usuários e aplicações enxergam dados publicados pelo *Le Select* como tuplas em tabelas relacionais, cujos dados não precisam residir em banco de dados: estes podem estar em arquivos textos, planilhas, etc.
Clientes que desejam usar dados e programas publicados conectam-se aos servidores correspondentes, através do componente cliente do *Le Select*, que colabora para o estabelecimento da conexão entre o componente cliente e o *site* de publicação. Clientes podem construir aplicações que aproveitam as características desse sistema, ou podem usar *Web browsers* para exploração *ad hoc* de dados e programas publicados.
O publicador precisa informar ao *Le Select* onde o dado reside e como acessá-lo. Essas informações são armazenadas em um módulo chamado tradutor (*wrapper*). *Wrapper* é uma coleção de classes *Java* que implementam uma Interface *Wrapper* [2]. Para cada fonte de dados é necessário um tradutor específico. Dessa forma um tradutor consegue ocultar a heterogeneidade das fontes de dados, apresentando-os para o *Le Select* no modelo relacional, podendo portanto serem acessados via SQL.
De modo similar, para prover acesso aos programas, o *Le Select* utiliza-se de *Wrappers* de programas, que correspondem aos módulos que implementam a lógica de execução de um programa, representado em XML. A figura 1 apresenta um exemplo de arquivo de definição de um tradutor de programa do tipo texto denominado "*LSTransformation.wd*". Esse arquivo é utilizado pelo tradutor para chamar a execução de transformações.

```
<ProgramWrapper
WrapperClass="lstransformation.MaterializerProgramWrapperFactory">
        <Parameters >
                <input Directory="/tradutores"  />
        </Parameters>
</ProgramWrapper>
```

Figura 1: Exemplo de um Wrapper de Programa

O *Wrapper* exemplificado nessa figura é escrito em XML, e é inicializado pela *tag ProgramWrapper*, indicando tratar-se de um *wrapper* de programa. O primeiro comando é o que faz a chamada ao *Program Wrapper Factory* do *Le Select*, sendo este o responsável pela identificação dos parâmetros que serão passados pelo usuário e pela execução da transformação desejada. O parâmetro de entrada passado nesse exemplo (*input Directory*) indica em qual diretório o *wrapper* de programa está armazenado.
Considere o exemplo da figura 2 no qual uma transformação é invocada. Nesse exemplo a função soma é chamada para somar 273,15 a uma coluna Temperatura70 relativa à temperatura em graus Celsius, com o objetivo de transformá-la em graus Kelvin.
O resultado da consulta JOB EXECUTE será uma tabela contendo uma coluna do tipo VARCHAR, constituída por uma linha. Este valor corresponde ao identificador do *job* que pode ser usado em qualquer outra chamada ao *Le Select*.
Os programas usados no *Le Select* podem residir em um servidor *host* enquanto os dados a serem processados podem estar em outra máquina. O *Le Select* envia os dados a serem manipulados para o *site* onde se encontra o programa (através do comando *Job*), coordena a execução e envia os resultados ao cliente sob forma de tabelas, assim como todos os demais dados acessados pelo *Le Select*. Após fazer uso dos resultados o sistema descarta-os, liberando espaço para futuros processamentos.

```
job execute /WrappersDePrograma/LSTransformation
parameter A ='soma'
parameter B = '273.15'
input dataset is select Temperatura_Minima from
tempminima/temp1/Temperatura70
```

Figura 2: Chamada para Transformação de ^{o}C em ^{o}K

4. O papel do metadado no processo de transformações

No contexto de DW o metadado tem uma função primordial: construir, manter e gerenciar a informação sobre os dados existentes, incluindo também o registro de todas as transformações sofridas pelo mesmo desde a sua captura das fontes operacionais até a sua inserção no Warehouse. Metadado é um conceito genérico, mas a cada implementação utiliza-se de métodos e técnicas específicos. Estes dependem das necessidades de cada organização, dos recursos existentes e dos requisitos de interface do usuário. Ainda não existem padrões de aceitação geral para a catalogação dos metadados das diversas fontes operacionais. Portanto, a definição destes metadados baseia-se no software de DW selecionado para a execução de uma determinada tarefa.
O metadado funciona, de certo modo, como o coração do ambiente do DW. Criar definições de metadado completas e eficientes pode ser um processo demorado, mas quanto melhores as definições, melhor será a compreensão da comunidade de usuários.
É importante lembrar que os dados armazenados em um DW apenas são fiéis quando inseridos num contexto conhecido e bem especificado. Portanto, a documentação dos metadados é essencial à plena utilização dos recursos disponíveis. Por este motivo, a utilização de um padrão de metadados para essa especificação é de extrema relevância.

4.1 Common Warehouse Metamodel (CWM)

CWM é uma especificação que descreve a troca de metadados entre DW, negócios inteligentes, gerenciamento do conhecimento e tecnologias de portal [5]. Desenvolvido pela OMG (*Object Management Group*), o CWM tem tido forte apelo para tornar-se padrão pelo W3C. Contitui-se de um *framework* para representar metadados sobre fontes de dados, objetivo dos dados, transformações e análise, além de processos e operações que criam e gerenciam DW, provendo informações sobre o seu uso [16].
O metamodelo CWM consiste de vários sub-metamodelos para representar o metadado de *Warehouse,* comum às principais áreas de interesse de DW, conforme apresentado a seguir:

- **Recursos de Dados** - Inclui metamodelos para representar dados orientados a objetos, relacionais, registros, multidimensionais e recursos de dados em XML. No caso de dados orientados a objetos, o CWM reusa e depende dos fundamentos da UML;
- **Análise de Dados -** Inclui metamodelos que representam transformações de dados, OLAP (Online Analytical Process - Processo analítico On-line), data mining, visualização de informação e nomenclatura do negócio;
- **Gerenciamento de Warehouse -** Inclui metamodelos para representar processos de Warehouse e resultados de operações de Warehouse;
- **Fundamentos** - Inclui metamodelos para representar informações do negócio, tipos de dados, expressões, chaves e índices, tipos de mapeamento e desenvolvimento de software.

No contexto deste trabalho o principal enfoque é dado ao metamodelo de Análise de dados, já que este se refere às transformações de dados, detalhado a seguir.

4.1.1 Pacote de Transformações

O pacote de transformação contém classes e associações que representam metadados comuns de transformação usados em DW, cobrindo as transformações básicas entre todos os tipos de fontes de dados. Este pacote é projetado para habilitar a troca de metadados comuns entre ferramentas e atividades de transformação, assumindo a existência dos seguintes pacotes de representação para tipos de fontes de dados potenciais ou destino: Modelo de Objeto (orientação a objeto), Relacional, Registro, Multidimensional, XML, OLAP, e *Data Mining*. O Pacote de transformação integra ainda os seguintes pacotes: OLAP, *Data Mining*, Processo de *Warehouse* e Operação de *Warehouse*. Em particular, os pacotes de transformação e Processo de *Warehouse* juntos provêem um construtor do metamodelo que facilita a programação e execução no DW. Os pacotes de Transformação e Operação de *Warehouse* juntos provêem um construtor do metamodelo que habilita a linearidade dos dados no DW.
Em cada passo da transformação é executada uma única tarefa de transformação. Esses passos se agrupam em atividades de transformação. Dentro de cada atividade a seqüência de execução de seus passos é explicitamente

definida por usar a dependência de precedência de passo ou precedência de restrição, ou implicitamente através da dependência de dados, como pode ser observado na figura 3.

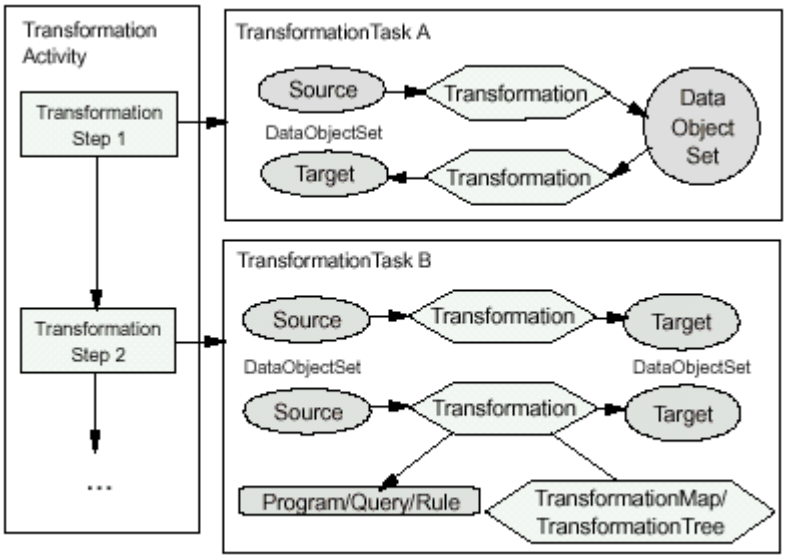

Figura 3: Pacote de Transformação [16]

Uma atividade transformação transforma um conjunto de objetos fonte em um conjunto de objetos destino. Os elementos de um conjunto de objetos podem ser quaisquer elementos do modelo de objeto, porém tipicamente representam tabelas, colunas ou elementos do modelo que representam os objetos transitórios na memória. Conjunto de objetos podem ser fonte e destino para diferentes transformações. Em particular, um determinado conjunto de dados pode representar o destino de uma transformação e a fonte de uma ou mais transformações dentro da mesma unidade lógica. Este é normalmente o caso da transformação que produz e consome objetos temporários.
O CWM permite ainda a descrição dos seus dados em XML de forma a viabilizar o intercâmbio de metadados do DW com outras ferramentas. A próxima seção mostrará o desenvolvimento dos *wrappers* de programas desenvolvidos para atuarem como serviços do *Le Select* no servidor, assim como sua descrição usando esse padrão.

5. Desenvolvimento de Wrappers de Programa

Conforme comentado na seção 3, um *wrapper* ou tradutor de programa é uma coleção de classes *Java* que implementam uma interface. Nosso objetivo no escopo deste trabalho é prover o *Le Select* de serviços específicos capazes de serem chamados de qualquer cliente *Le Select* para realizarem transformações em dados extraídos de fontes publicadas na *Web*.
A figura 4 apresenta um esboço de atuação do *Wrapper*, mostrando como este se comportará no contexto de transformação de dados.
Considere uma consulta na qual uma operação de junção é realizada sobre as tabelas S1(A,B) e S2(B,D), constituídos pelos atributos A, B e D, conforme apresentado na figura 4. Nesse exemplo, o usuário passa os parâmetro A e D pertencentes às bases S1 e S2 respectivamente. O *Wrapper* do programa correspondente a esta transformação recebe os parâmetros da consulta, e tendo em vista que ele conhece as especificações dessas bases (o *Le Select* acessa o *Wrapper* de dados dessas bases que contém todas as especificações necessárias, tais como: nome da tabela, número de colunas e o tipo de cada uma, URL, etc), ele tem como realizar uma junção com o atributo B comum às duas tabelas. A partir daí, tem-se uma única tabela, que é convertida para o formato exigido (ou seja, traduzir para uma tabela relacional, caso ela não esteja neste padrão). O serviço de transformação é então executado e os resultados que são acessados através do tradutor de dados, são retornados como resposta ao usuário. Para melhor entender a utilização do *wrapper*, considere que S1(A,B) corresponde à tabela Temperatura70 (mencionada na figura 2), p corresponde à implementação da operação Soma e p.wrapper corresponde ao *Wrapper* de programa *LSTransformation*. O usuário passa como consulta ao *Le Select* o comando *job* visto na figura 2. O *Le Select* recebe esses parâmetros, acessa a base de dados especificada e reúne todos estes dados, que são passados junto com a chamada ao *LSTransformation*, que acessa e executa p. Estes novos resultados são repassados para o *Le Select* que os disponibiliza aos usuários após colocá-los no padrão de um *wrapper* de dados.

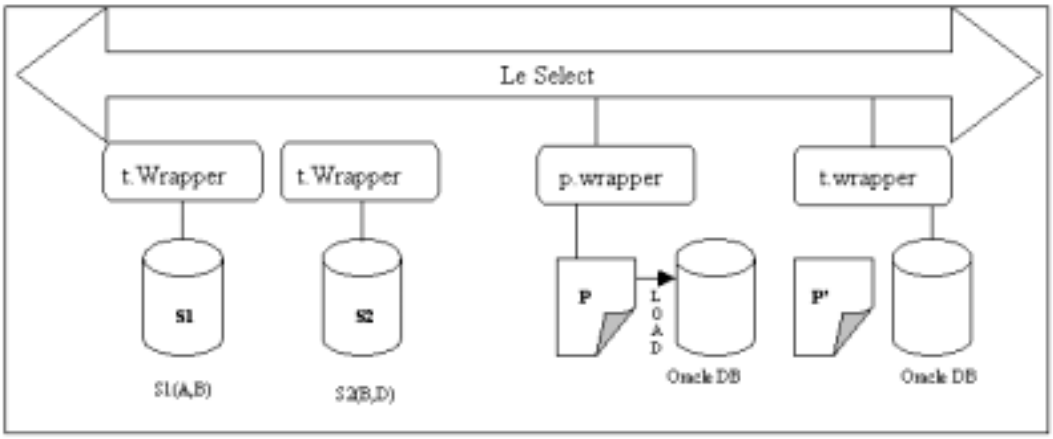

Figura 4: Utilização do Wrapper de Programa

5.1 Etapas de Transformação

A seguir são descritos alguns aspectos sobre o desenvolvimento dos *wrappers* de programas que representam, na verdade, um módulo de implementação da lógica de execução de um programa (no caso, as transformações) no *Le Select*. Esses wrappers, implementados em *Java*, têm como função proporcionar a transformação dos dados publicados pelo sistema *Le Select* antes que estes sejam carregados na área de organização de dados de um DW.
O diagrama de classes em UML mostrado na figura 5 apresenta as classes principais para criação desses *Wrappers*, com suas respectivas descrições.

- **ProgramWrapperFactory:** Essa classe cria e configura uma instância do *wrapper* de programa. O arquivo de configuração do *wrapper* indica o nome da classe do *Program Wrapper Factory*, além de passar as informações das configurações para a criação do *wrapper*, cujo método principal é:
    - **getWrapper(InfoNode):** o parâmetro *InfoNode* obtém as informações necessárias providas do objeto de configuração do wrapper e as passa para o objeto construtor.
- **ProgramWrapper:** A tarefa dessa classe é executar o programa para os determinados dados e parâmetros de entrada, tornando o resultado disponível para ser visualizado por um wrapper de dados. Possui os seguintes métodos:
    - **newInstance():** Cria uma instância, representando uma invocação do programa. Depois de criada, a instância será chamada para executar o programa.
    - **restoreInstance(stateObject):** Cria uma instância que restabelece seu estado através do argumento stateObject. O stateObject pertence a um ProgramInstance do mesmo tipo e será salvo durante (ou depois de) sua execução.
- **ProgramWrapperInstance:** Quando um programa é executado, um objeto PragramWrapperInstance é criado. Este objeto controla a invocação do programa, recebe dados e parâmetros de entrada e executa o programa. Após a execução do programa ele cria os wrappers de dados para permitir que os resultados sejam acessados. Possui os métodos:
    - **runPass1(rs, directory, p, saveNotifier):** Corresponde ao primeiro passo de execução do programa. Neste método são passados como parâmetros o conjunto de resultados e as propriedades (que são parâmetros passados na consulta do usuário) que serão de fundamental importância na execução das transformações. Neste método também são realizadas as chamadas para a classe Materializer. Se o programa falhar (ou for parado pelo cancel(), antes de sua conclusão), uma exceção deverá ser lançada;
    - **runPass2():** Corresponde ao segundo passo da execução do programa. Retorna os nomes das tabelas do resultado, tal como encontrado dentro de cada wrapper. Se o programa falhar (ou for parado pelo cancel(), antes de sua conclusão), uma exceção deverá ser lançada;
    - **getResultsWrapperDefinition():** Esse método é chamado após o método runPass2() ou após uma recuperação. Retorna os conteúdos de um ou mais arquivos de definição de wrappers que serão usados pelo *Le Select* para criação de um ou mais wrappers que acessem os resultados. Desde que os resultados estejam disponíveis (qualquer run() ou restore() deverá retornar OK), este método não pode resultar em erro (nenhuma exceção pode ser lançada);
    - **dispose():** Remove qualquer dado produzido pelo programa. Deve ser exigido também após uma recuperação.

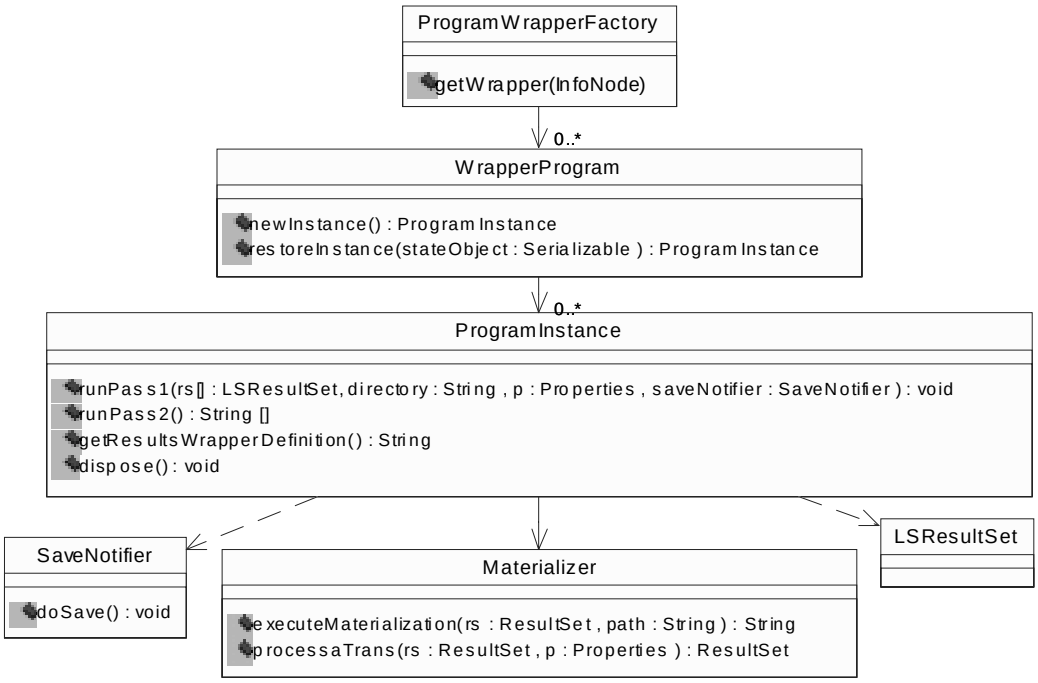

Figura 5: Diagrama de Classes

- **Materializer:** Essa classe é responsável por implementar parte do processo de execução de consultas assíncronas. Seus principais métodos são:
    - **executeMaterialization(rs, path):** Realiza a materialização dos dados, acessando um conjunto de resultados (rs) e armazenando-os em um arquivo temporário. Após realizar a transformação nos dados este método é chamado para materializar os resultados;
    - **processaTrans(rs, p):** Executa a transformação dos dados. Estas podem ser: Conversão de formatos, aplicação de faixas de intervalos (intervalo, faixaEtaria), transformações de *string* para *string* (tfmString, sexo), transformações de *string* para double (tfmStringEmDouble),etc.

O método **processaTrans** é de grande importância, pois é nele que estão implementados os tipos de transformações que podem ser realizadas. A seguir o seu código será quebrado em partes para se ter uma melhor compreensão do seu funcionamento.

```
public ResultSet processaTrans (ResultSet rs, Properties p) throws WrapperException,IOException,SQLException{
```

Este método recebe como parâmetros: **rs:** corresponde a um conjunto de resultados, equivalentes ao resultado da consulta feita pelo usuário, e **p:** corresponde aos parâmetros passados pelo usuário.

```
int Ncol = meta.getColumnCount();
String param1 = p.getProperty("A");
```

Ncol recebe o número de colunas contidas no conjunto de resultados originados da consulta do usuário. O param1 equivale ao primeiro parâmetro passado pelo usuário, que corresponde ao tipo da transformação que será realizada.

```
if (param1.equalsIgnoreCase("soma"))
```

Nesta parte do código inicia-se a execução de cada transformação inserida em param1. No exemplo em questão este corresponde a transformar o valor de uma coluna pela soma da mesma a um valor. Porém, este poderia ser qualquer outro processo de transformação implementado. Os códigos variam segundo as operações embutidas em cada transformação. Devido a limitações de espaço, apresentamos nessa seção um exemplo de transformação simples, porém que inclui pontos em comum às diversas transformações implementadas no *Le Select*. O comando *equalsIgnoreCase* é utilizado para que não haja distinções entre letras maiúsculas e minúsculas.

```
if (Ncol==1)
```

A partir do momento em que o param1 foi reconhecido (no exemplo, soma), entra-se na execução do seu código. Assim, se o número de colunas for igual a 1, isso indica que existe uma coluna que será somada a algum valor.

```
while(crset.next()){
double coluna = crset.getDouble(1);
int param2 = Integer.parseInt(p.getProperty("B"));
coluna = coluna + param2;
crset.updateDouble(1,coluna);
```

Esta parte do código faz com que cada linha da coluna seja somada ao valor passado pelo usuário através do parâmetro param2.

```
else{
while (crset.next()){
double coluna1 = crset.getDouble(1);
double coluna2 = crset.getDouble(2);
coluna2 = coluna1 + coluna2;
crset.updateDouble(2,coluna2);}
```

Caso o número de colunas seja diferente de 1, então 2 colunas deverão ser somadas. Estas colunas devem ser do mesmo tipo e devem ter o mesmo número de linhas. A coluna a ser atualizada será a segunda a ser passada por consulta pelo usuário.

5.2 Uso do CWM para Descrição das Transformações

Com o objetivo de padronizar o processo de transformações esta seção mostra a descrição das mesmas utilizando o padrão de metadados CWM. Essa catalogação é importante, pois irá facilitar ao usuário o entendimento de como estas são realizadas, permitindo a completa descrição dos dados e metadados correspondentes a estas transformações, e viabilizando um possível intercâmbio desses códigos com outras ferramentas.
De modo a demonstrar o uso do CWM no processo de transformação, apresentaremos a seguir o mesmo exemplo da seção anterior, voltado para a conversão de escala, associada a uma coluna (Temperatura70 é somada a um valor).
O Documento XML a seguir refere-se à SOMA (coluna com valor), cujo processo faz parte das operações básicas de transformação incluídas no *Le Select*.

```
<?xml version="1.0"?>
<DOCTYPE  transformation SYSTEM “transformation.dtd”>
<documento>
        <Transformation>
        <!--SOMA-->
                <TransformationActivity>
                <!--Somar coluna com valor-->
                        <TransformationStep>
                        <!--Informar o primeiro parametro-->
                                <TransformationTask Param1="soma"/>
                        </TransformationStep>
                        <TransformationStep>
                        <!--Obter numero de colunas-->
                                <TransformationTask Ncol="1"/>
                        </TransformationStep>
                        <TransformationStep>
                        <!--Obter resultado da consulta-->
                                <TransformationTask rs="coluna"/>
                        </TransformationStep>
                        <TransformationStep>
                        <!--Obter segundo parametro-->
                                <TransformationTask Param2="valor"/>
                        </TransformationStep>
                        <TransformationStep>
                        <!--Executar soma-->
                                <TransformationTask  coluna2="coluna+Param2"/>
                        </TransformationStep>
                </TransformationActivity>
        </Transformation>
</documento>
```

Todo o processo de transformação descrito segundo esse padrão é armazenado no próprio *wrapper* de programa do *Le Select*, na sua parte relativa à documentação.

6. Conclusão

Neste trabalho foi proposta a utilização do uso conjunto das tecnologias de mediação e DW com vistas ao desenvolvimento posterior de um DW ambiental.
Uma importante contribuição desse trabalho é a disponibilização das principais transformações normalmente realizadas por ferramentas ETL, e que antecedem a fase de carga de dados num DW. Essas transformações foram

disponibilizadas como serviço do *middleware Le Select*, para o qual foram desenvolvidos tradutores (*Wrappers*) de programas. Com o objetivo de possibilitar o intercâmbio dessas transformações com outras ferramentas foi utilizado o padrão de metadados CWM, com forte tendência de ser recomendado pela W3C, para a descrição de processos em DW e armazenado no próprio *wrapper* do *Le Select*.

Como trabalho futuro pretendemos dar prosseguimento ao desenvolvimento de um DW Ambiental (DWA), constituído por dados extraídos da *Web* por este middleware e transformados utilizando o serviço criado no escopo deste trabalho.

Outro ponto importante a ser investigado diz respeito à integração de fontes *Web* que, conforme já evidenciado na literatura [6][14] oferece um grau de complexidade bem maior do que a integração de banco de dados federados e distribuídos [17] [1]. Isso decorre do fato de que estas fontes além de serem heterogêneas e distribuídas são dinâmicas, podem ser estruturadas ou não, e suas descrições nem sempre estão disponíveis. Além disso, como o projetista do DWA não tem nenhum controle sobre esses dados, nem sempre será possível atestar quanto a fidelidade dos mesmos.

Referências

[1] Amit, P. e Sheth, James e Larson, A., "Federated Database System for Managing Distributed, Heterogeneous, and Autonomous Databases", ACM Computing Surveys, Vol 22, 3, 1990.

[2] Amzal, M. e Manolescu, I. e Simon, E. e Xhumari, F. e Lavric, A., "*Sharing Autonomous and Heterogeneous Data Sets and Programs with Le Select''*, http://caravel.inria.fr/Fprototype_LeSelect.html.

[3] Arantes, Álisson R. e Laender, Alberto H.F. e Golgher, Paulo B. e Silva, Altigran S. da, "*Managing Web Data Through Views*". Proceedings of the Second International Conference, EC-Web, Monique, Alemanha, Setembro 2001.

[4] Bergamaschi, S. e Castano, S. e Vincini, M., "*Semantic Integration of Semistructured and Structured Data Sources*", Sigmod Record, Vol20, n^0 1, Março 1999.

[5] Cover, Robin, "*OMG Common Warehouse Metadata Interchange Specification*", Setembro de 2000.

[6] Dittrich, K. e Domenig, R., "Towards Exploitation of the Data Universe", 3rd International Conference on Business Information System, Abril 1999.

[7] Dittrich, K. e Domenig, R., "*An Overview and Classification of Mediated Query Systems"*. Sigmod Record vol 28, 3, Set. 1999.

[8] *IBM Almaden Research Center*, especificação do projeto Garlic, 1995, http://www.almaden.ibm.com/cs/garlic/homepage.html.

[9] Inmon, W. H., "*Building the Data Warehouse*", John Wiley&Sons, 1996.

[10] Kimball, Ralph, "*Data Warehouse Toolkit*", São Paulo, Makron Books, 1998.

[11] Kimball, Ralf e Reeves, Laura e Ross, Margy e Warren Thornthwaite, "*The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses*", John Wiley & Sons Inc., New York, 1998.

[12] Konig-Ries, B. e Reck, C., "*An Architecture for Transparent Access to Semantically Heterogeneous Information Sources*". Proceedings of the 1st International Workshop on Cooperative Information Agents, Berlim, 1997.

[13] "*Le Select: a Middleware System for Publishing Autonomous and Heterogeneous Information Sources*". INRIA, França, 1999, obtido no site: http://www.rodin.inria.fr/~xhumari/LeSelect/index.html.

[14] Marotta, A. e Motz, R. e Ruggia, R. "*Managing Source Schema Evolution in Web Warehouses"*. International Workshop on Information Integration on the Web - Technologies and Applications (WIIW), Rio de Janeiro, Brasil, abril 2001.

[15] Martinez, M.R. e Roussopoulos, N., "*MOCHA: A Self-Extensible Database Middleware System for Distributed Management Data*". Proceedings of The ACM Sigmod International Conference On Management Data, Dallas, Texas, USA, 2000.

[16] OMG Document, "*Common Warehouse Metamodelo (CWM) Specification*", 02 de Fevereiro de 2001, obtido no site http://www.omg.org.

[17] Özsu, M.T. e Valduriez, P., "*Principles of Distributed Database Systems*", Prentice-Hall, 1999.

[18] Singh, Herry, "*Data Warehousing"*, Prentice-Hall, 1999.

[19] Wiederhold, G., "*Mediation to Deal with Heterogenous Data Sources"*. In: Proceedings of the INTEROP'99. Zurich, 1999. http://hake.stanford.edu/pub/gio/.