UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ELSON RODOLFO DE MELO

UM AMBIENTE EXTENSÍVEL DE APOIO À CONSTRUÇÃO DE FERRAMENTA ETL

Recife
2004

ELSON RODOLFO DE MELO

UM AMBIENTE EXTENSÍVEL DE APOIO À CONSTRUÇÃO DE FERRAMENTA ETL

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de mestre em Ciência da Computação. Área de concentração: Banco de Dados

Orientador: Dr. Décio Fonseca

Coorientador: Dr. Roberto Souto Maior de Barros

Recife

2004

Catalogação na fonte
Bibliotecária Luiza de Oliveira, CRB4-1316

M528u Melo, Elson Rodolfo de
Um ambiente extensível de apoio à construção de ferramenta ETL / Elson Rodolfo de Melo. – 2004.
143 f.: il.

Orientador: Décio Fonseca.
Coorientador: Roberto Souto Maior de Barros

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIN, Ciência da Computação, Recife, 2004.

Inclui referências.

1. Data Warehouse. 2. Ferramenta ETL. 3. ambiente desenvolvimento extensível. I. Fonseca, Décio (orientador). II. Barros, Roberto Souto Maior de (Coorientador). III. Título.

025.04 CDD (23. ed.) UFPE - CCEN 2022-147

ELSON RODOLFO DE MELO

UM AMBIENTE EXTENSÍVEL DE APOIO À CONSTRUÇÃO DE FERRAMENTA ETL

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Pernambuco, Centro de Informática, como requisito para a obtenção do título de Mestre em Ciência da Computação. Área de concentração: Banco de Dados.

Aprovado em: 27/02/2004.

BANCA EXAMINADORA

_____

Prof. Dr. Roberto Souto Maior de Barros (Orientador)
Centro de Informática - UFPE

_____

Prof. Dr. Fernando da Fonseca de Souza (Examinador Interno)
Centro de Informática - UFPE

_____

Prof. Dr. Marcus Costa Sampaio (Examinador Externo)
Departamento Sistemas e Computação - UFCG

Dedico este trabalho a minha família, meu principal pilar e meu maior tesouro, pela compreensão dos momentos ausentes, pela colaboração constante nas profícuas opiniões.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, pois sempre acreditaram na minha capacidade e sempre apostaram no meu sucesso, não lhes importando os obstáculos ou as dificuldades que eu enfrentava. Devo a eles boa parte do que hoje sou e espero continuar aprendendo todos os dias com os seus ensinamentos e atitudes.

Uma dissertação de mestrado é um trabalho realizado com a participação de pessoas que convivem com o mestrando durante a sua pesquisa e contribuem de diversas formas para que ele consiga chegar ao objetivo final que é a defesa de sua tese. Por isto, eu agradeço a todos os amigos, colaboradores, e companheiros de percurso que direta ou indiretamente contribuíram para que este trabalho terminasse, com a qualidade e conteúdo alcançados.

Agradeço ao professor Décio Fonseca, meu orientador, pessoa que me apoiou continuamente nesta empreitada e foi fundamental na conclusão deste trabalho. Sem a sua participação, este sonho não teria sido possível. O professor Décio é um amigo há vários anos e influenciou decisivamente a minha carreira. O considero um modelo de profissional e pessoa honrada, justa e competente.

Dedico especial obrigado a Senhora Ivoneide da Silva Ribeiro que sempre me apoiou nos momentos mais confusos, apresentando-me caminhos e orientando-me de forma adequada e dedicada nas ações administrativas referentes ao meu curso de mestrado.

Agradeço a Tânia Elizabethe, minha esposa e parceira, que me apoiou e incentivou nos momentos fundamentais desta jornada.

Agradeço finalmente a Deus, provedor de todo o bem, pelo milagre da vida que nos concede diariamente, pela energia que nos permite superar as dificuldades, concedendo-nos a inteligência humana com a qual temos a aptidão de desfrutar as Suas Dádivas e encontrar a alegria e felicidade na vida.

Fecho afirmando que ninguém é tão grande que não possa aprender, nem tão pequeno que não possa ensinar. Só tolos acreditam que tudo já sabem. Neste trabalho aprendi e amadureci com as pessoas com quem interagi e suas ideias. E ofereço a elas a minha gratidão pelos seus ensinamentos e experiências que servirão para minha vida.

RESUMO

A maioria das organizações tem feito uso da tecnologia para controlar os seus negócios. Isto tem implicado num crescente número de sistemas transacionais que estão gerando um grande volume de dados. Devido à necessidade de uma melhor gestão, as organizações estão sendo forçadas a se tornarem mais competitivas, o que as têm levado a explorar o potencial informacional destes seus dados operacionais, extraindo-lhes informações estratégicas e gerenciais que estão sendo armazenadas em Data Warehouse e Data Marts. Durante o ciclo de desenvolvimento destes Data Warehouses, a etapa que consome maiores esforços e recursos é a implementação do processo ETL, do acrônimo em inglês Extract, Transform and Load. Por isto, muitas organizações estão optando pelo uso de ferramentas específicas, projetadas para auxiliar o desenvolvimento do processo de povoamento dos Data Warehouse. O fato é que estas ferramentas são muitas vezes onerosas e nem sempre atendem a todos os requisitos definidos para o processo ETL e por isto a necessidade de se criar um ambiente mais flexível e capaz de fornecer uma solução ajustada à realidade de cada projeto. Este trabalho visa apresentar um ambiente extensível capaz de oferecer um framework *open source* para a construção de ferramentas ETL. Este ambiente permite a implementação de funcionalidades voltadas ao processo ETL e a incorporação de novas estruturas e componentes de forma colaborativa e ordenada, criando um ciclo de desenvolvimento evolutivo e compartilhado. O material foi montado de forma que os primeiros textos exibem os principais conceitos sobre Data Warehouse. Em seguida, é apresenta uma pesquisa onde foram levantados os aspectos avaliativos a serem adotados pelas organizações durante a aquisição de ferramentas ETL e mostradas as principais características de algumas ferramentas disponíveis atualmente. A dissertação termina descrevendo o ambiente de apoio à construção de ferramentas ETL, através de sua arquitetura, suas funcionalidades e seus aspectos evolutivos.

Palavras-chave: Data Warehouse; ferramenta ETL; ambiente desenvolvimento extensível.

ABSTRACT

Most organizations have made use of technology to control their business. This has implied an increasing number of transactional systems that are generating a large volume of data. Due to the need for better management, organizations are being forced to become more competitive, which has led them to explore the informational potential of their operational data, extracting strategic and managerial information from them that are being stored in Data Warehouse and data. marts. During the development cycle of these Data Warehouses, the step that consumes most efforts and resources is the implementation of the ETL process, which stands for Extract, Transform and Load. For this reason, many organizations are opting for the use of specific tools, designed to help the development of the Data Warehouse population process. The fact is that these tools are often expensive and do not always meet all the requirements defined for the ETL process and therefore the need to create a more flexible environment capable of providing a solution adjusted to the reality of each project. This work aims to present an extensible environment capable of offering an open-source framework for building ETL tools. This environment allows the implementation of functionalities aimed at the ETL process and the incorporation of new structures and components in a collaborative and orderly manner, creating an evolutionary and shared development cycle. The material has been assembled in such a way that the first texts show the main concepts about Data Warehouse. Then, a research is presented where the evaluative aspects to be adopted by organizations during the acquisition of ETL tools were raised and the main characteristics of some tools currently available are shown. The dissertation ends by describing the support environment for the construction of ETL tools, through its architecture, its functionalities, and its evolutionary aspects.

Keywords: Data Warehouse; ETL tool; extensible development environment.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

BMD	Bancos multidimensionais
CRM	Gerenciamento de Relacionamento com Cliente
DM	Data Mart
DW	Data Warehouse
ERP	Planejamento de Recursos Empresariais
ETL	Extract,Transform, Load
IDE	Integrated Development Environment
MOLAP	Multidimensional On-Line Analytical Processing
ODS	Operational Data Store
OLAP	On-Line Analytic Processing
ROLAP	Relational On-Line Analytical Processing
SAD	Sistema de Apoio a Decisão
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A área de Data Warehouse, (DW), é relativamente recente em Banco de dados se levarmos em consideração outras, como modelagem de BD, SGBDs relacionais. No entanto, ela já ganhou muita importância e dimensão devido ao grande número de aplicações e sistemas desenvolvidos, e ao crescente número de pesquisas na área. Os sistemas de DW estão sendo criados na maioria das organizações e a tecnologia empregada tem evoluído continuamente, tanto em relação a software como a hardware. Atualmente, existe uma grande quantidade de aplicativos direcionados para DW e a capacidade de armazenamento e recuperação de dados dos equipamentos disponíveis não para de crescer.

Além da evolução tecnológica, o surgimento de empresas de informática especializadas na área tem estimulado o crescimento do número de Data Warehouses implementados. Algumas agregam valor aos seus serviços fornecendo ferramentas e utilitários.

Hoje, as empresas e corporações entendem que precisam converter a grande quantidade de dados operacionais, obtida no processamento diário de seus sistemas, em informações gerenciais, que serão estratégicas e servirão de base no processo de tomada de decisões. Por isto estão investindo consideravelmente em DW, implicando diretamente no crescimento das empresas especializadas na área que têm acelerado a evolução desta tecnologia.

1.1 MOTIVAÇÃO

Uma das etapas mais complexas da construção e manutenção de um Data Warehouse é a obtenção, transformação e carga de seus dados (ETL). O grande desafio está no fato de que os dados para alimentação do DW estão distribuídos entre vários sistemas, se encontram em formatos diversos e a identificação do que é relevante é um problema mais gerencial e administrativo do que técnico. Além disto, uma vez identificado e entendido o que é importante, os procedimentos para obtenção dos dados são muitas vezes complexos e podem envolver muitos métodos, técnicas e ferramentas. A fase de coleta, transformação e carga dos dados é a mais longa e a de maior risco em todo o processo de implementação de um DW, podendo chegar a até 80% de tempo e custo do projeto (CAMPOS, FILHO, 1997).

Não existe atualmente uma solução definitiva e consolidada para a realização do povoamento dos DW. Até pelas características peculiares do processo como heterogeneidade das fontes de dados, sistemas legados em plataformas e linguagens diferentes, existe a necessidade de um padrão que atenda a todas as nuances do processo. Por isto o mercado apresenta soluções geralmente compostas por equipes de profissionais e ferramental específico, uma configuração montada com consultores, analistas, programadores, apoiados por ferramentas proprietárias de suas empresas ou fornecidas por terceiros, que normalmente automatizam, facilitam ou ajudam a acelerar as etapas de extração, transformação e carga – ETL do DW. Analisando o mercado de empresas, nacionais e estrangeiras, constata-se que as ferramentas de ETL, muitas vezes nasceram a partir de um projeto de construção de um DW que realizaram, onde a expertise adquirida no processo foi ajustada e preparada para ser empregada em outros projetos semelhantes.

Em relação às ferramentas usadas na construção e manutenção de DW, é importante que elas sejam bem avaliadas não apenas pelas suas funcionalidades, mas também pela sua capacidade de tratar os principais problemas do processo ETL. Além de existirem muitas soluções no mercado, às vezes são necessárias duas ou mais delas para que todas as fases do processo sejam atendidas. Seus custos são normalmente elevados e, geralmente, são vendidas junto com consultorias e treinamentos compulsórios.

Algumas grandes empresas como Microsoft® e Oracle® fornecem utilitários com uma grande abrangência de soluções para a realização de povoamento de DW. O problema delas é que suas soluções são voltadas para o seu ambiente, são pouco flexíveis e direcionadas para os seus bancos de dados, restrição relevante para quem deseja independência de plataforma ou produto. Em contrapartida, seus produtos são bastante completos, testados e possuem o amadurecimento peculiar das grandes fornecedoras de soluções prontas, pacotes, na área de informática. Este fato, a médio e longo prazo, pode reduzir o investimento do cliente e diminuir os riscos inerentes ao processo de desenvolvimento e implantação.

Finalmente, apesar de existirem muitas ferramentas ETL disponíveis no mercado, nenhuma delas pode ser considerada como a solução completa para o problema de povoamento de DW. Isto porque a natureza do problema é complexa e cheia de nuances, normalmente envolvendo bases de dados, sistemas e aplicativos heterogêneos. Sempre surgirão novas ferramentas capazes de apoiar na solução do

problema, algumas delas construídas nas próprias organizações que as utilizam, como produtos internos.

1.2 TRABALHOS CORRELATOS

Alguns trabalhos apresentam aspectos do processo de povoamento de DW que estão diretamente relacionados com o tema desta dissertação. O GRUPO DATAAWARE (2003) da Universidade Federal do Rio de Janeiro, disponibiliza artigos e teses referentes a DW, alguns deles ligados ao processo ETL, como o elaborado por PEREIRA (2000), “Uso do Padrão OIM de Metadados no Suporte às Transformações em Ambiente de Data Warehouse”. Neste trabalho, Denise apresenta uma proposta de extensão ao padrão OIM, usando o DTS, um dos produtos apresentados nesta dissertação como modelo de ferramenta ETL disponível, como base para a sua dissertação. O seu trabalho focou na extensão de classes e interfaces usadas no processo ETL, que em certo grau de relação, é defendida no ambiente proposto nesta tese. Outro trabalho correlacionado é apresentado por André Fernandes da Costa (COSTA, FELIPE, 2001), que focou nos aspectos relacionados a criação e manutenção de DW, dando ênfase ao projeto físico.

Em outra linha de associação, estão os trabalhos ligados ao REDIRIS que é um ambiente integrado de construção de sistemas de informação. O REDIRIS, descrito no anexo A, estabelece um ambiente aglutinador de tecnologias, no qual a ferramenta proposta nesta dissertação se integra. O material descritivo sobre o REDIRIS aqui apresentado foi obtido a partir do trabalho desenvolvido por SANTOS (2002). O ambiente Hydrus definido nesta dissertação se integra ao REDIRIS fornecendo a infraestrutura para implementação do núcleo do ambiente, especificamente o que trata os componentes: captura e análise, integrador, pré-processamento e construtor de soluções. As implementações do ambiente Hydrus devem ser feitas segundo os princípios básicos que regem a concepção e construção do REDIRIS, descritos no anexo A.

1.3 OBJETIVO DO TRABALHO

1.3.1 Objetivo principal

Este estudo visa, uma vez que não existe uma solução definitiva para o problema de extração, transformação e carga de dados em Data Warehouse, apresentar o ambiente Hydrus, com características especiais, usado no apoio à construção de ferramentas ETL de forma ordenada e através da colaboração entre programadores.

O ambiente Hydrus, direcionado para o processo de povoamento de DW, incorpora conceitos evolutivos como plugins e scripts, além de registrar de forma natural as estruturas e regras definidas para as ferramentas ETL em metadados armazenados em bancos de dados. A capacidade de harmonizar a coexistência entre a ferramenta ETL definida e a implementação dos seus processos ETL faz do ambiente Hydrus um repositório centralizado, compacto e prático.

A simplicidade do Hydrus é reconhecidamente a qualidade que o torna viável, pois permite o seu uso sem conhecimentos profundos dos usuários e garante o seu emprego em projetos de pequeno e médio porte. Embora não ofereça complexas funcionalidades nativas, o ambiente permite a implementação delas a partir do esforço de seus usuários, atendendo a demanda sobre medida, ao longo do processo de desenvolvimento.

Como a criação de ferramentas ETL que atendam as reais necessidades dos projetos, sem grandes investimentos, é inviável, o modelo de desenvolvimento usado no código livre foi escolhido para o ambiente Hydrus, valorizando a nova onda que graça no universo do desenvolvimento de software e aproveitando a capacidade de produção que esta filosofia traz, isto é, através do trabalho colaborativo e ordenado.

Entende-se por código livre, do termo em inglês *free software*, a capacidade dos usuários de um software poderem executá-lo, estudá-lo, copiá-lo e distribuí-lo, e modificá-lo, sem quaisquer ônus. São, portanto, quatro tipos de liberdade que o usuário possui em relação ao produto classificado como código livre:

- A liberdade de executar o programa, sem restrições, para qualquer propósito;
- A liberdade para estudar como o programa funciona e adaptá-lo às suas necessidades. Para isto, o acesso ao código-fonte é um pré-requisito obrigatório.

- A liberdade para redistribuir cópias do programa, sem ônus de qualquer espécie, sem necessidade de pagamento de licenças ou custos de aquisição.
- A liberdade para alterar o software e distribuir as alterações para a comunidade, de maneira que todos possam se beneficiar das mudanças. Acesso ao código-fonte é um pré-requisito para que isto aconteça.

Embora a proposta pareça ambiciosa, o foco em um ambiente simples e pequeno para a construção de ferramentas ETL, mas voltada à extensão pode ser a alternativa ideal as grandes ferramentas onerosas e com excesso de funcionalidades. O Hydrus, com suas classes, usando tecnologias atuais como XML pode ser uma ótima opção para os pequenos e médios projetos.

1.3.2 Objetivos Secundários

Além da proposta de uma ferramenta ETL, este trabalho também tem como objetivo:

- Caracterizar, em linhas gerais, o processo de Data Warehousing, com seus aspectos e singularidades, enfocando o processo de extração, transformação e carga de dados nos DW.
- Apresentar um estudo sobre ferramentas de apoio ao processo ETL de DW, apresentando suas características principais e contextualizando-as no trabalho, e listar algumas destas ferramentas que são consideradas referência pela qualidade alcançada e funcionalidades oferecidas.

1.4 ORGANIZAÇÃO DA TESE

Além deste capítulo introdutório, onde foram apresentados os objetivos da pesquisa e o cenário no qual ela se situa, este trabalho compreende os seguintes capítulos:

- Capítulo 2 – Ambiente de Data Warehouse: neste capítulo são apresentadas as principais características de um DW, sendo analisados os conceitos gerais como arquitetura, metadados e modelagem dimensional.

- Capítulo 3 – Povoamento de Data Warehouse: este capítulo analisa o processo ETL detalhando suas principais características, focando nos problemas que ocorrem durante a execução da atividade de povoamento de DW e apresentando soluções adotadas.
- Capítulo 4 – Ferramentas ETL: este capítulo foca nas ferramentas ETL, apresentando um estudo contendo os seus principais aspectos, abordando as suas características mais relevantes que devem ser analisadas pelas organizações no momento de adquiri-las. Ele também apresenta as ferramentas ETL mais conhecidas atualmente, mostrando seus pontos positivos e as funcionalidades oferecidas aos usuários, fechando com um breve comparativo entre elas.
- Capítulo 5 – Ambiente Hydrus: este capítulo fecha a dissertação, apresentando o ambiente com suas principais estruturas e classes. São detalhados os principais aspectos do Hydrus que são corroborados com a apresentação de um estudo de caso real, uma descrição de uma primeira implementação realizada em linguagem de desenvolvimento Delphi.
- Capítulo 6 – Considerações Finais: este capítulo contém conclusões sobre o assunto apresentado além de propostas para trabalhos futuros.

2 AMBIENTE DE DATA WAREHOUSE

2.1 INTRODUÇÃO

O uso crescente da tecnologia está fazendo com que o número de sistemas em operação cresça e com eles a quantidade de dados armazenados, que detém, implicitamente, informações sobre o negócio ao qual estão relacionados. A reunião de forma coerente destes dados transformando-os em informações gerenciais, que possam ser utilizadas nas análises e usadas nas tomadas de decisões das organizações, é o caminho para a definição de um Data Warehouse.

Normalmente, as organizações dependem da experiência de seus funcionários para determinar a direção para onde estão indo. Isto pode ser um fator de risco para elas, pois estão dependentes da visão subjetiva que as pessoas têm dos seus negócios e a experiência que possuem pode ser perdida na reestruturação do seu quadro de pessoal. Por isto, a importância do DW que armazena as informações sobre o negócio independentemente e de forma perene e ininterrupta.

As organizações geram dados diariamente durante as suas atividades operacionais. No entanto, na maioria delas, o conteúdo deste material não está disponível para consultas que ajudem a gerência a entender o seu negócio. Este cenário é desalentador, pois existe um enorme potencial em termos de informação nestes dados. Uma das abordagens para mudar esta situação é a construção de um Data Warehouse onde todo o conteúdo gerado pelos sistemas em operação possa ser integrado e tratado, formando um imenso banco histórico que retrate a vida da organização.

2.2 CARACTERÍSTICAS

Quando se está projetando um Data Warehouse, deve-se estar atento às suas peculiaridades de forma que o projeto contemple requisitos apropriados e alcance os resultados esperados. Sua construção envolve uma análise baseada no entendimento das expectativas dos usuários e na identificação do que é relevante para ser incluído em sua base de dados. Algumas características inerentes ao DW não podem ser ignoradas durante a elaboração do seu projeto, podendo ser citadas algumas delas:

- Base de dados classificada por assunto, integrada e preparada especialmente para a realização de consultas;
- Base de dados sem atualizações contínuas, mas em intervalo de tempo pré-determinado;
- SGBD configurado adequadamente para trabalhar sobre um grande volume de dados;
- Sistemas projetados para realização de consultas ad hoc e voltados à gerência;

As características citadas anteriormente são apenas algumas das muitas que o DW possui. Deste ponto em diante, far-se-á um estudo das principais, muitas das quais são apresentadas no conceito definido por INMON (1997): Granularidade, tempo de atualização e retenção de dados no DW, escalabilidade, disponibilidade, integração, não volatilidade, orientação por assunto, qualidade de dados e metadados.

Pela importância do tema, neste trabalho serão discutidos de forma sucinta a integração, qualidade dos dados e metadados.

2.2.1 Integração

Os dados do DW são obtidos, normalmente, de várias bases operacionais, algumas até desativadas. Uma das características do DW é a integração correta destas bases para formar o seu banco de dados, através da análise e execução de operações de extração, transformação e carga.

Aparentemente simples, a integração é uma característica complexa porque envolve muita análise e trabalho para ser realizada corretamente. Existem muitos problemas que dificultam uma boa integração que crie uma base de dados com qualidade. Por exemplo, as diferenças e variações semânticas que existem nos dados dos sistemas transacionais. É o caso das idades das pessoas que são registradas diferentemente nos sistemas: como valores absolutos (28, 30, ...,100 anos) ou por faixas (10 a 20, 21 a 30, ..., mais de 70 anos). A solução para este problema está nas técnicas de transformação, fase do processo ETL nos DW.

Independente dos problemas que possam existir, a integração correta é uma característica primordial para o DW, pois ela garante a qualidade dos seus dados.

Falhas de integração geram dados que não servem às consultas, pois fornecerão informações erradas aos gerentes, levando-os a tomar decisões equivocadas.

2.2.2 Qualidade dos dados

A qualidade dos dados de qualquer sistema influencia diretamente o grau de confiabilidade que ele apresenta. Quando a base possui dados incorretos, geralmente provenientes de algum processo de entrada executado com falhas, as informações extraídas dela não são confiáveis e não podem ser usadas. Gerentes que tomam decisões sobre dados de má qualidade estarão arriscados a cometer equívocos que podem causar grandes danos ao seu negócio.

Muitos projetistas investem bastante trabalho na especificação de relatórios complexos e que atendam às necessidades dos usuários, mas esquecem que, se os dados de onde serão obtidas as informações não tiverem qualidade, estarão disponibilizando relatórios inúteis, que não servirão aos propósitos para os quais foram criados. Por exemplo, suponha que um sistema de vendas registre a forma de pagamento dos clientes. Se porventura houver erro nestes registros, o valor do caixa disponível que a organização acredita ter não estará correto o que trará problemas diversos. A má qualidade dos dados também pode levar a informações absurdas que normalmente não são percebidas de imediato. Por exemplo, há histórias de relatórios médicos que apresentam percentual maior do que zero de pessoas grávidas do sexo masculino.

Estabelecer normas e controles que garantam a qualidade dos dados nos sistemas de DW é primordial para se alcançar o sucesso. A lista a seguir apresenta algumas características que podem ser usadas para este fim:

Quadro 1 - Requisitos para determinar qualidade dos dados de Data Warehouse.

Requisitos	Descrição
Precisão	Mensura o quanto os dados estão corretos;
Abrangência	Grau que mede a relação entre a quantidade de informações requisitadas versus à atendida;
Consistência	Mede o grau de veracidade, firmeza e realidade dos dados;
Coerência	Mede o grau de congruência dos dados com as regras estipuladas para eles

Fonte: o autor (2004)

2.2.3 Metadados

Metadados pode ser definido como dados que descrevem dados, uma abstração dos dados ou ainda como dados de alto nível que descrevem dados de um nível inferior. Qualquer uma destas definições pode ser utilizada. O importante é entender que os metadados dão significado aos dados, explicando como eles estão organizados, como podem ser compartilhados e como podem ser utilizados para os diversos fins.

Para o Data Warehouse, os metadados são muito importantes. Um bom projeto de DW deve prever a implementação de vários metadados, cada um servindo de suporte a uma categoria de atividades sobre os seus dados. A inexistência de metadados integrados capazes de descrever completamente os dados dificulta ainda mais a integração e o compartilhamento dos dados nas organizações (BRACKETT, 1996).

Os metadados representam a figura integradora dos vários níveis que compõem a arquitetura de informação do DW (INMON, HACKATHORN, 1994). Além disto, os metadados, que davam suporte apenas aos programadores de software e gerentes de SGBDs nos sistemas transacionais, passaram, nos DWs, a ser úteis para o usuário final, os analistas de negócio. Para um ambiente de Data Warehouse, o metadado é crucial, diferente do ambiente transacional onde ele pode ser considerado opcional (INMOM, ZACHMAN, GEIGER, 1997).

Fazendo-se uma análise sobre metadados, pode-se agrupá-los em três camadas diferentes:

- Metadados operacionais: guardam informações sobre os dados operacionais mantidos pelos sistemas transacionais;
- Metadados centrais do DW: armazenam informações sobre os dados do DW, tais como as transformações sofridas no processo ETL, agregações existentes, campos calculados, visões definidas.
- Metadados do nível do usuário: guardam informações sobre os dados do Data Warehouse em formato apropriado, que pode ser compreendido e utilizado pelo usuário de negócio.

Os metadados nos DW, diferentes dos pertencentes ao ambiente transacional, precisam ser mantidos ao longo do tempo, mesmo que sofram alterações. Na realidade, deve haver a manutenção de todas as suas versões, metadados históricos,

pois os dados que eles descrevem, no Data Warehouse, duram ao longo do tempo (INMON, HACKATHORN, 1994) .

Uma das grandes dificuldades na manutenção dos metadados é a inexistência de um padrão e repositório comum para eles. Muitas das ferramentas utilizadas nos DWs mantêm seus próprios metadados, em formatos proprietários. Além disto, elas não conseguem entender os formatos uma das outras. Esta situação complica o trabalho de quem dar manutenção nas bases do Data Warehouse que se vê obrigado a trabalhar com diversos modelos de metadados. Adotar um padrão e criar um repositório central e compartilhado de metadados pode aumentar a produtividade e facilitar a manutenção e escalabilidade dos DWs

2.3 ARQUITETURA DO DATA WAREHOUSE

A definição correta da arquitetura a ser adotada para um Data Warehouse é essencial para o seu sucesso. Uma escolha errada pode determinar o fracasso de todo o projeto. A arquitetura determina os componentes que farão parte do DW e como eles estarão dispostos e inter-relacionados. Ela também define a forma como o DW será montado e influencia em muitas características vistas anteriormente como escalabilidade e disponibilidade. Hoje os problemas dos DW estão mais relacionados com a arquitetura adotada do que propriamente com a tecnologia escolhida (MELO, 1997).

Apresentar as estruturas que compõe um DW e o papel desempenhado por cada uma delas é uma das finalidades da arquitetura que deve ser adequada à organização. Muitos fatores influenciam na sua definição. Existem muitas arquiteturas propostas para um DW, cabendo ao projetista descobrir qual delas será melhor, mais ajustada ao seu problema e realidade.

2.3.1 Componentes comuns

Nas arquiteturas definidas para os DWs aparecem vários componentes com papéis específicos. Alguns são referentes ao armazenamento de dados, outros ao processamento de consultas. Nesta seção serão apresentados os mais relevantes a este texto, mostrados através de descrição rápida de suas características.

2.3.1.1 Repositórios

Os repositórios são entidades onde os dados são armazenados. Basicamente existem três tipos: Data Warehouse que será referenciado neste texto como RDW e Operational Data Store – ODS.

2.3.1.1.1 RDW

Pode-se dizer que o RDW é uma grande base que integra dados tratados e filtrados, provenientes das bases operacionais e fontes externas, com objetivo de guardar o conteúdo usado na construção de relatórios e consultas gerenciais que são destinadas a apoiar a tomada de decisão nas organizações. Este repositório é o maior dos três citados neste documento e contém informações sobre várias áreas da organização. Seus dados são mantidos armazenados por um longo tempo e são orientados a assunto. É importante que o conceito de RDW, que se refere simplesmente ao banco, não seja confundido com o de ambiente de Data Warehouse.

2.3.1.1.2 ODS – Operational Data Store

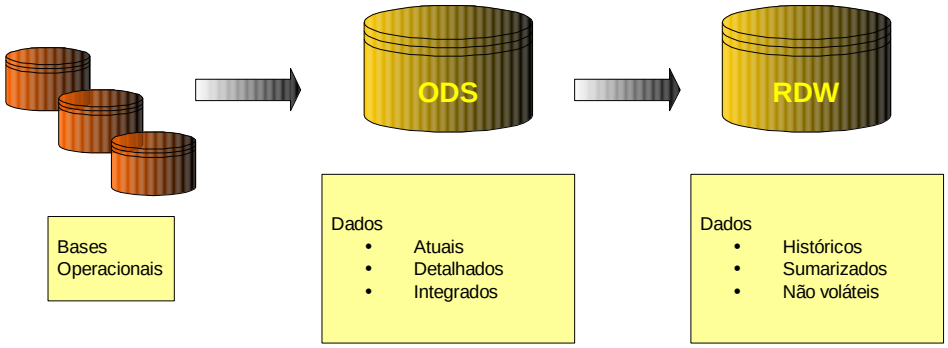
Os ODS armazenam dados do ambiente operacional antes deles serem incorporados ao DW. Estes dados são atualizados continuamente por um período determinado, pelo menos 24 horas, até estarem aptos a serem transferidos para o DW. Como se pode ver, o conteúdo dos ODS(s) é transitório.

Quando os dados são obtidos das bases operacionais, eles são integrados e tratados nos ODS(s) através de filtros e limpeza. Este tipo de procedimento evita que dados imaturos, ainda sem a qualidade necessária sejam utilizados para povoar o DW.

Uma das funções dos ODS(s) é garantir maior segurança ao ambiente de DW. Como são bases intermediárias, se ocorrer algum problema enquanto estão sendo atualizadas, falhas que as tornem inconsistentes ou indisponíveis, não haverá comprometimento do funcionamento do DW.

A figura 1 mostra uma representação do ODS e RDW, contextualizando-os e citando as principais diferenças existentes entre eles:

Figura 1 - Diferença entre ODS e Repositório Data Warehouse



Fonte: o autor (2004)

2.3.2 Arquiteturas

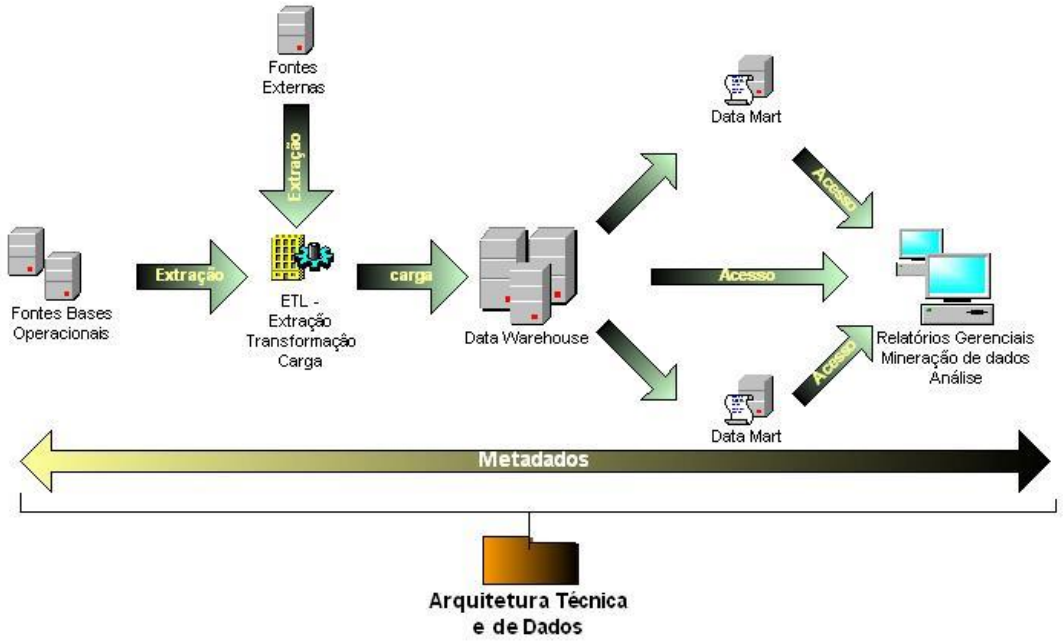
Definidos alguns conceitos e componentes que existem no ambiente de DW, tem-se o conhecimento necessário para se entender algumas das arquiteturas disponíveis e usadas nas organizações. Em todas elas a forma como são compreendidas determina a maneira como são montadas e usadas, além dos componentes que são desenvolvidos. As arquiteturas mais simples podem ser usadas como solução em um maior número de projetos e por isto são mais universais e aceitas. Quando a arquitetura fica mais complexa ou específica, ou mesmo assume uma abordagem mais singular, ela se torna restrita, sendo empregada em menor número de projetos de DW.

2.3.2.1 Uma visão de uma Arquitetura geral

Uma arquitetura de um ambiente de DW simplificada pode ser formulada a partir de um conjunto de hardware, software, padrões e serviços. Nela têm-se as estruturas de armazenamento, mecanismos de integração, comunicação, processamento e apresentação das informações aos usuários finais.

Ela pode ser subdividida em quatro partes composta por: arquitetura de dados, arquitetura técnica, administração e gerenciamento do DW, além dos metadados. A figura 2 mostra uma visão geral desta arquitetura.

Figura 2 - Visão geral de arquitetura de DW



Fonte: o autor (2004)

2.3.2.1.1 Arquitetura de dados

A arquitetura de dados descreve quais os componentes de armazenamento do DW serão construídos e como estarão inter-relacionados. Ela apresenta a estrutura de armazenamento, as fontes de onde foram obtidos, os modelos físicos e lógicos adotados na sua concepção, além das agregações e hierarquias existentes.

A arquitetura de dados define a granularidade, volume e distribuição dos dados no ambiente de Data Warehouse (KIMBALL, REEVES, ROSS, THORNTHWAITE, 1998).

2.3.2.1.2 Arquitetura técnica

A arquitetura técnica compreende os processos e ferramentas que atuam sobre os dados. Ela define como e quais são as atividades executadas sobre o DW para que os dados estejam disponíveis para o usuário final. Também faz parte da arquitetura técnica a definição dos recursos computacionais necessários ao funcionamento e manutenção do DW.

A arquitetura técnica é subdividida em duas partes: *back room* e *front room*. A primeira é responsável pelo povoamento do DW. Ela compreende as atividades de

extração, transformação e carga dos dados do DW obtidos das bases dos sistemas transacionais. O *front room* compreende a interface entre o DW e seus usuários. Ela é responsável por fornecer as informações que a gerência precisa, através da utilização de ferramental apropriado e hardware.

2.3.2.1.3 Administração e gerenciamento de DW

A administração e gerenciamento de DW é o componente responsável por manter o DW atualizado e consistente. É sua responsabilidade manter a infraestrutura do DW funcionando, dando o suporte necessário. Além disto, as seguintes atividades também são de sua alçada: segurança, obtenção e tratamento de dados, qualidade de dados, atualização de metadados, monitoramento de performance, gerenciamento e retirada de dados caducos desnecessários, backup e recovery, replicação e distribuição de dados.

2.3.2.1.4 Metadados

O último componente da arquitetura geral de um DW é o metadado, que mantém informações sobre os dados armazenados. Ele é utilizado tanto no processo de construção como no de obtenção de informações do DW.

Nas próximas seções, serão apresentadas arquiteturas mais específicas, disponíveis no mercado e aceitas como padrão. Nem todas as arquiteturas existentes foram colocadas neste trabalho, apenas as mais comuns. Algumas que podem ser encontradas em outros textos são variações destas que estão aqui apresentadas.

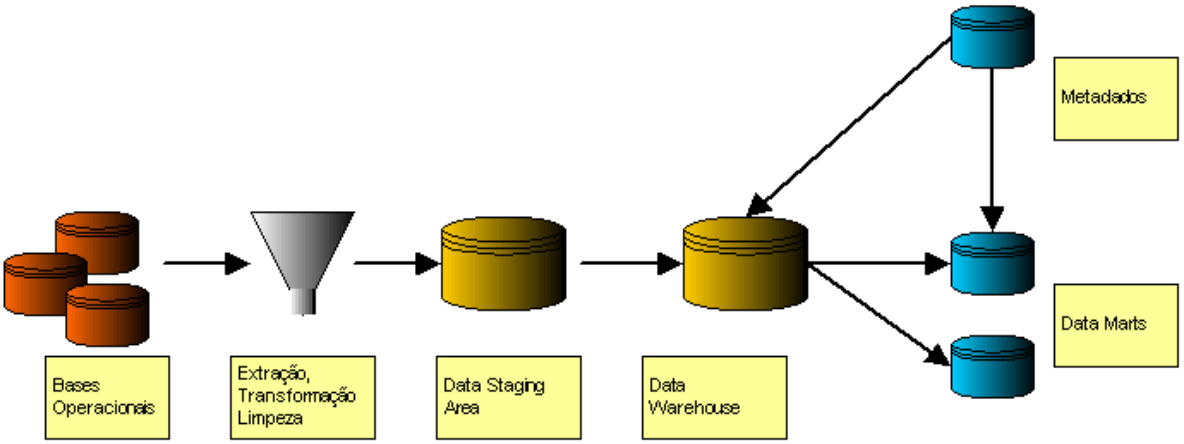
2.3.2.2 Arquitetura Top-Down

A arquitetura top-down é a mais conhecida atualmente e tem-se tornado padrão em Data Warehouse. Ela foi concebida por INMON (1997) e contém os seguintes componentes de dados:

- Área de preparação de dados – data staging área
- Repositório Data Warehouse
- Data Marts
- Metadados

A figura 3 mostra como estes componentes são organizados e como é montada a arquitetura.

Figura 3 - Visão geral de arquitetura de DW



Fonte: (COSTA, FELIPE, 2001)

A ideia principal da arquitetura top-down é armazenar os dados que são obtidos dos sistemas transacionais em uma base intermediária, chamada de Data Staging Area, onde são tratados e preparados para povoar o DW. Este processo é executado pelas aplicações ETL. Depois de tratados, os dados contidos na área intermediária são carregados no Data Warehouse que servirá como fonte para os Data Marts.

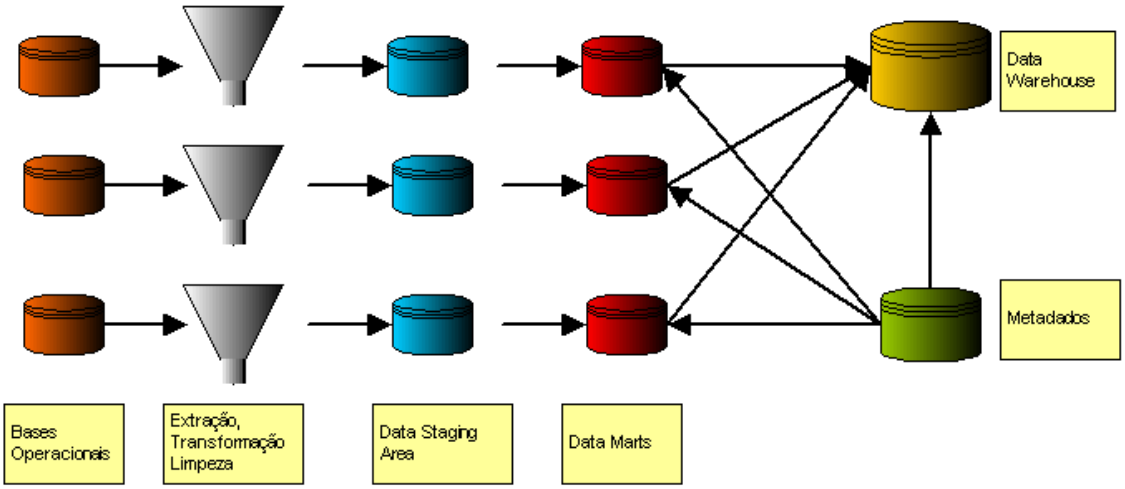
Cada data mart obtém seus dados do DW que possui baixa granularidade e alta retenção de informações, para poder fornecer o conteúdo com a riqueza de detalhes necessária. Os DMs armazenam os seus dados em bases dimensionais que são usadas pelas ferramentas de mineração e geradoras de relatórios para fornecer ao usuário as informações pedidas. Nesta arquitetura, não há consultas gerenciais realizadas sobre o DW diretamente.

A grande vantagem desta arquitetura é a qualidade dos dados armazenados nos DMs, pois eles são obtidos diretamente do RDW que mantém uma base bem montada e centralizada. A desvantagem desta arquitetura é o longo tempo gasto na sua implementação que aumenta os custos e os riscos do processo. Por causa disto, muitas organizações não a adotam.

2.3.2.3 Arquitetura Bottom-Up

A arquitetura Bottom-Up usa os Data Marts como ponto de partida para a construção do Data Warehouse, conforme é apresentado na figura 4. Nela, os DMs são povoados com os dados obtidos diretamente das bases operacionais, através das operações de extração, transformação e carga. Esta arquitetura também prevê o uso de Data Staging Área, durante a etapa de povoamento das bases de dados dos DMs.

Figura 4 - Arquitetura Bottom-Up



Fonte: (COSTA, FELIPE, 2001)

Os dados do DW, na arquitetura Bottom-up, são obtidos dos DMs. Este fato causa muitos problemas, pois cada DMs pode possuir dados em formatos diferentes, que precisam ser tratados e integrados antes de serem gravados no DW. Além disto, cada DM mantém os seus próprios metadados, também em formatos proprietários e isto impede a existência de um repositório central e compartilhado de metadados.

A arquitetura Bottom-Up apresenta resultados imediatos, já que ela não obriga a construção de todos os DMs previstos no projeto para que DW entre em operação. Por isto ela é apreciada pelas organizações, pela rapidez com que o produto entra em produção. Mas esta vantagem pode ser um problema em longo prazo, pois as diferenças geradas durante a proliferação desordenada de DM costumam tornar o processo de integração das bases no DW oneroso ou impossível. A solução para se evitar isto, é a definição, antes da criação dos DMs, de padrões para os modelos de

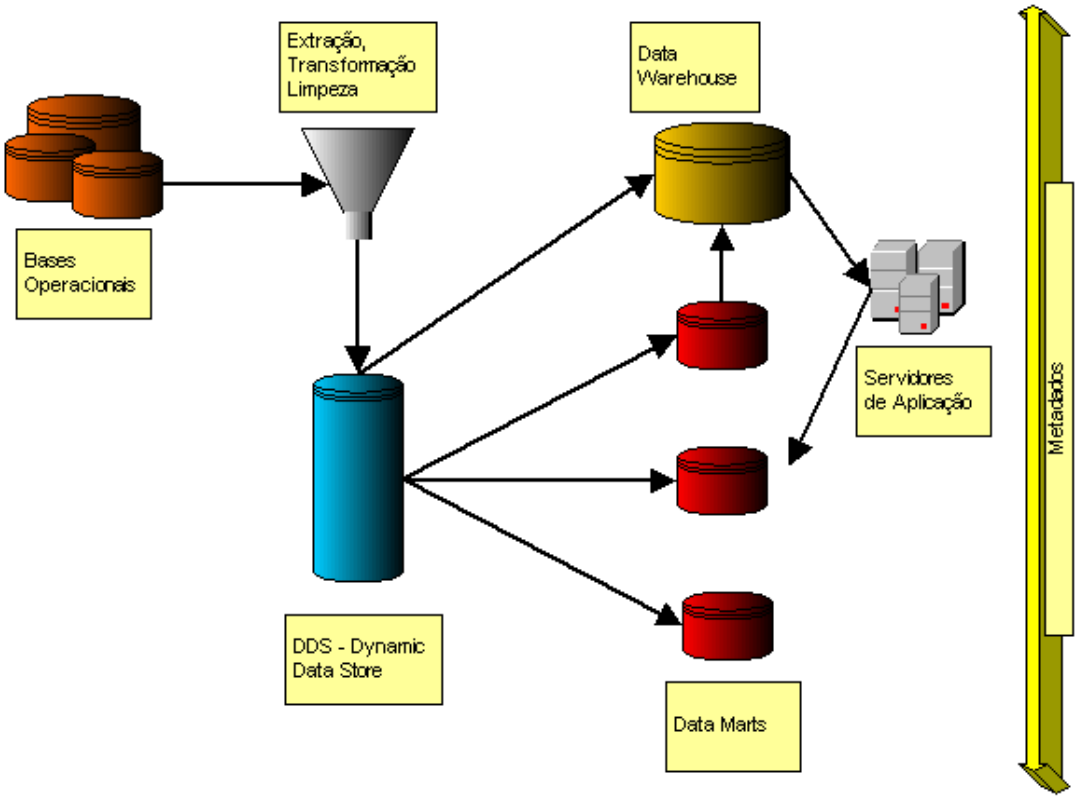
dados, metadados, softwares, entre outros, que viabilizem o crescimento ordenado do DW.

Arquiteturas derivadas da Bottom-Up estão surgindo com mecanismos que impedem o colapso resultante do crescimento desordenado e sem padrões de DMs.

2.3.2.4 Arquitetura EDMA – Enterprise Data Mart Architecture

A arquitetura Bottom-Up apresenta problemas quando o número de DM aumenta consideravelmente. Para acabar com esta limitação foi criada uma arquitetura derivada chamada de EDMA. Nela se define e utiliza-se um framework compartilhado na construção dos Data Marts, como é mostrado na figura 5.

Figura 5 - Arquitetura EDMA



Fonte: (COSTA, FELIPE, 2001)

O objetivo do framework é fornecer ordem e padronização ao processo de construção dos DMs. Ele consegue isto estabelecendo métricas, definindo regras de negócios comuns, e compartilhando os metadados no DW. Sua implementação é feita através de dois repositórios de dados: o DDS – *Dynamic Data Store* e o repositório global de metadados.

O DDS é uma área de preparação dos dados, comum a todo o ambiente de DW, onde o conteúdo proveniente do ambiente operacional é tratado e transformado, ficando prontos e disponíveis para os DMs. Ele é altamente volátil, estando em constante atualização.

O repositório global de metadados é compartilhado por todo o ambiente do DW, promovendo, desta forma, consistência semântica e padronização.

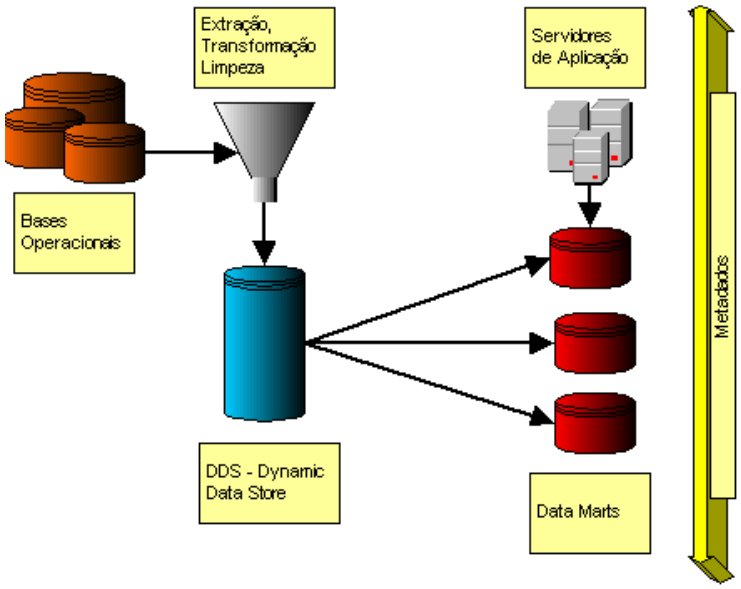
A arquitetura EDMA representou um grande avanço na construção de Data Warehouse.

2.3.2.5 Arquitetura DS/DM – Data Stage/Data Mart Architecture

Como uma variação da arquitetura EDMA, surge a DS/DM. Praticamente idênticas, a grande diferença entre elas é a inexistência do repositório do Data Warehouse na arquitetura DS/DM, que é substituído pela conjunção dos DMs disponíveis.

Aparentemente mais simples, a arquitetura DS/DM tem uma enorme desvantagem em relação a EDMA, pois a ausência do DW centralizado dificulta ou impossibilita a obtenção de uma visão ampla do negócio. Para conseguir informações gerais, precisa-se varrer as bases dos DM que são limitadas e só apresentam determinado assunto da organização. A figura 6 mostra esta arquitetura.

Figura 6 - Arquitetura DS/DM



Fonte: (COSTA, FELIPE, 2001)

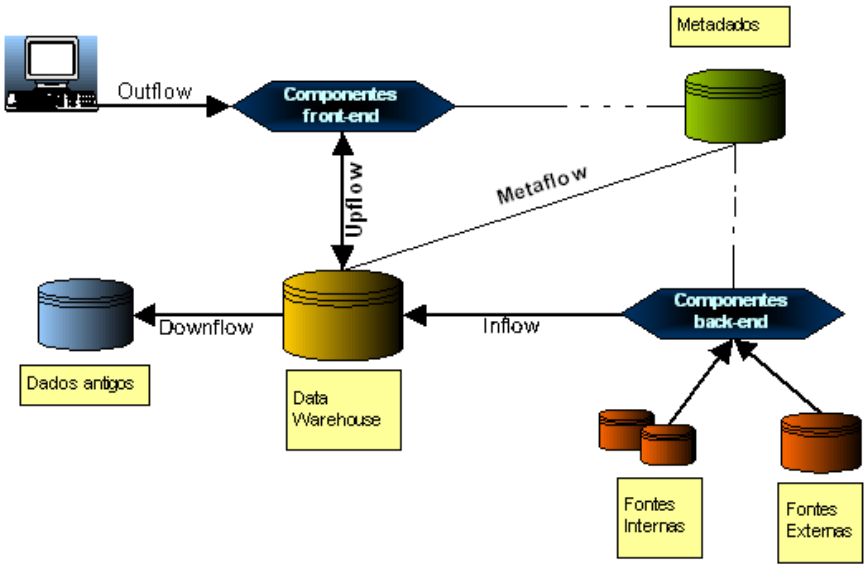
2.3.2.6 Arquitetura Segundo Chaudhuri

A arquitetura segundo CHAUDHURI (CHAUDHURI, DAYAL, 1997) é baseada na compreensão dos fluxos de dados que ocorrem no sistema de DW. “O verdadeiro valor de um sistema de DW não está em apenas colecionar dados, mas sim, gerenciar fluxos de dados” (HACKATHORN, 1997).

Para esta arquitetura, o ambiente de DW é examinado através da origem dos dados e o seu fluxo, apresentando uma abordagem diferente das mostradas anteriormente neste documento. Ela é organizada em:

- Fontes que fornecem os dados ao DW: bases operacionais e fontes externas de dados;
- Um conjunto de estruturas de dados analíticos armazenados: o DW do sistema;
- Um Sistema Gerenciador de Banco de Dados (SGBD) otimizado para atender as requisições analíticas dos sistemas de DW;
- Um componente *back end*: conjunto de aplicações responsáveis por extrair, filtrar, transformar, integrar e carregar os dados de diferentes origens no DW;
- Um componente *front end*: conjunto de aplicações responsáveis por disponibilizar as informações aos usuários finais do DW;
- Um repositório para armazenar e gerenciar os metadados do sistema.

Figura 7 - Arquitetura segundo Chaudhuri



Fonte: (MORAIS, 1998)

Os cinco principais fluxos que fazem parte desta arquitetura são: entrada (*inflow*), saída (*outflow*), subida (*upflow*), descida (*downflow*) e o metaflow (*metaflow*).

O fluxo de entrada (*inflow*) é responsável pelo povoamento do DW através do processo ETL, onde os dados, oriundos dos sistemas operacionais (internos) e externos, são extraídos, filtrados, transformados, integrados e carregados no DW. O fluxo de descida ou *downflow* ocorre sempre que o DW precisa retirar de sua base os dados que não são mais úteis, considerados antigos (INMON, 1996). O fluxo de subida (*upflow)* prepara os dados para que eles possam estar em formato adequado e disponível para o usuário. Este fluxo fornece dados diretamente do DW para as aplicações *front-end.* Além disto, o *upflow* também tem a função de fornecer os dados do DW para os DMs e bancos de dados pessoais que geralmente estão localizados em estações de trabalho. O penúltimo fluxo é o de saída (*outflow*) que ocorre quando os dados são consultados pelos usuários que precisam de informações em formato apropriado e claro. O quinto e último fluxo é o *metaflow* que, ao contrário dos anteriores que explicam como dados se movimentam, determina como os metadados são transportados.

2.4 MODELAGEM DE DADOS

Vários autores definiram modelos de dados que podem ser usados em DW. Na verdade, o modelo tem papel fundamental no desenvolvimento das estruturas do banco, pois ele fornece um nível de abstração que o usuário consegue entender e trabalhar, definindo suas soluções. A modelagem permite a definição das estruturas dos dados do Data Warehouse. Ela consegue representar a realidade tratada no projeto, organizando os objetos e entidades que compõe o problema, mapeando-os para uma representação no banco de dados.

Para realizar a modelagem, normalmente se usa um ferramental especializado tais como o ER-Win® e o Rational Rose®. Tais produtos facilitam o trabalho do projetista fornecendo ambientes gráficos onde é possível modelar visualmente o banco de dados e gerar automaticamente a documentação e os scripts a partir das definições criadas.

Em DW, a modelagem precisa ser direcionada a pesquisa, a extração de informação. Modelos utilizados nos ambientes operacionais, apesar de serem utilizados em alguns projetos de DW, não são adequados, porque trabalham com

entidades cuja disposição, relacionamento e restrições são próprios para as funções dos sistemas transacionais.

Quando se está analisando um negócio, precisa-se de uma visão dos dados sobre uma perspectiva diferente da encontrada no mundo operacional. O foco principal, neste caso, é o assunto e o objetivo a consulta. O modelo mais aceito para este tipo de ambiente, de Data Warehouse, é o dimensional e suas variações.

No próximo capítulo será tratada a questão mais central deste trabalho que é o povoamento de DW.

3 POVOAMENTO DE DATA WAREHOUSE

3.1 INTRODUÇÃO

O povoamento de um Data Warehouse é tido por muitos especialistas como uma das operações mais complexas, que envolve maior quantidade de recursos computacionais. Além disto, a construção do ambiente que permite a sua execução consome muito esforço dos projetistas e programadores de sistemas, equipe técnica que precisa avaliar bem o problema e definir as soluções adequadas para que o povoamento seja executado corretamente.

O processo que executa o povoamento do DW é conhecido pela sigla ETL, acrônimo para o termo em inglês *Extract, Transform and Load*. Para que ele seja realizado precisa-se de um ambiente especialmente projetado composto de ferramentas e equipe especializada, além de equipamentos apropriados.

A maioria das organizações, principalmente aquelas de pequeno porte, que está criando o seu primeiro DW, têm implementado o processo ETL de forma artesanal e baseado na experiência do seu grupo de profissionais responsável pela tarefa. São equipes que tendem a usar as linguagens de desenvolvimento que conhecem, como Cobol, Pascal, PL/SQL, para construir os programas que serão usados no povoamento do DW. Outras organizações diferentemente adotam ferramentas prontas e direcionadas ao processo ETL, além de contratar consultores especializados no mercado. Na verdade, a solução mais adequada depende muito do tamanho, perfil e objetivos da organização que está implementando o Data Warehouse.

O processo ETL é batizado por alguns especialistas como operações de *back end*. Ele é disparado em períodos constantes ou executado continuamente, alimentando o DW com informações obtidas dos sistemas transacionais e fontes externas. Os dados nesta atividade são extraídos das diversas origens, tratados e incorporados ao DW de forma que a base resultante tenha a qualidade e conteúdo desejado pelos usuários finais.

Povoar um Data Warehouse sem o suporte de ferramentas especializadas é possível, porém não recomendável. Existem muitas no mercado fornecidas por diversas organizações. Elas apoiam várias das atividades que ocorrem no processo ETL e podem ser agrupadas de acordo com suas funções (IDG, 1998):

- *Extração, filtragem, transformação e migração*: estas ferramentas são responsáveis por obter os dados dos sistemas transacionais e, depois de transformá-los, carregá-los no DW. Geralmente são necessárias mais de uma ferramenta para realizar todas as etapas do trabalho.
- *Transferência de dados e replicação*: estas ferramentas apoiam atividades que podem ser consideradas como um complemento da extração. Elas transferem dados de uma base para outra, sem tratá-los. O uso delas ocorre em situações específicas tais como: replicação de dados, transporte de dados de um DMs para um RDW ou vice-versa, criação de cópias usadas como backup.
- *Gerenciamento e administração*: neste grupo estão as ferramentas utilizadas na gerência do processo ETL. Elas monitoram o comportamento do DW, apresentando informações sobre performance e segurança. Embora não sejam essenciais estas ferramentas dão maior controle aos administradores sobre o processo de povoamento do Data Warehouse.

Os processos ETL precisam ser bem projetados e implementados. Eles vão determinar o tipo e qualidade de conteúdo disponibilizado no DW. Para esclarecer mais este assunto, neste capítulo serão mostradas as alternativas disponíveis para execução destes processos.

3.2 PROCESSO ETL

O processo ETL é considerado como a fase mais complexa do ciclo de vida do Data Warehouse. Alguns fatores podem ser apresentados como justificativa para esta afirmação:

- Variedade de fontes e bases que precisam ser acessadas durante a execução do processo;
- Transformações complicadas ou trabalhosas que são feitas sobre os dados das bases operacionais;
- Grande volume de dados tratados no processo;

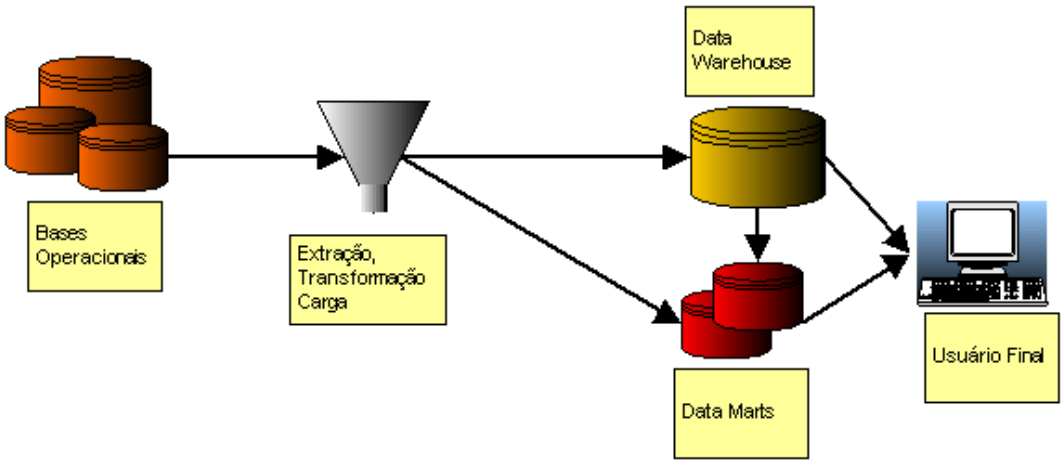
Como o processo ETL é empregado na manutenção dos dados do DW, ele precisa ser executado sempre que as fontes forem atualizadas. Ele consome uma grande quantidade de recursos computacionais e dependendo da forma como foi

implementado precisa ser disparado e monitorado por operadores. Cabe aos projetistas definirem a maneira como o processo será executado e quais os mecanismos e software que serão empregados. Por exemplo, eles precisam resolver se vão utilizar no processo ferramentas produzidas por terceiros. Alguns autores acham que o desenvolvimento de programas para o processo ETL só é justificável quando existe apenas uma única fonte de dados. Se múltiplas origens precisam ser acessadas durante o processo, é recomendada a adoção de ferramental apropriado, adquirido no mercado (CRAIG, VIVONA, BERCOVITCH, 1999).

Através de uma visão simplista do processo ETL, pode-se definir as seguintes etapas apresentadas dentro do ambiente de DW conforme a figura 8:

- *Extração:* os dados são obtidos dos sistemas transacionais, bases de dados legadas, e fontes externas como planilhas, relatórios, arquivos textos. São selecionados e armazenados em uma área temporária.
- *Transformação:* é aplicado sobre os dados obtidos na extração, tratamento que compreende geração de chaves, conversão de tipos, formatação de campos, união de tabelas, normalização e derivação de valores.
- *Carga:* os dados devidamente preparados são transferidos da área temporária para o DW, ficando disponíveis para consulta.

Figura 8 - Processo ETL de um ambiente de DW



Fonte: o autor (2004)

3.2.1 Opções para construção de uma solução para o processo ETL

Muitas são as alternativas para implementação de uma solução para o processo ETL. Todas vão depender de como o problema está apresentado e quais os objetivos

e recursos disponíveis pela organização. Didaticamente, pode-se dividir as soluções em quatro categorias:

- Desenvolvimento do software
- Utilização de um gerador de código
- Emprego de uma ferramenta ETL
- Adoção de uma solução híbrida

3.2.1.1 Desenvolvimento do software

A priori, pequenas organizações não desejam investir na aquisição de produtos que apoiem o processo ETL porque eles são caros. Por isto, elas usam a suas equipes de programadores na codificação de suas próprias soluções ou terceirizam o trabalho. Quando seguem esta linha, os gerentes de desenvolvimento precisam decidir que linguagem adotar. Se, porventura, a organização está trabalhando com bases relacionais, o mais indicado é a programação em linguagens baseadas em SQL, como a PL/SQL da Oracle®. Mas se as fontes de dados são heterogêneas, linguagens de desenvolvimento universais, como C++ ou até Cobol serão normalmente mais adequadas.

Outro aspecto do problema é a forma como a solução é implementada. Por exemplo, KIMBALL (KIMBALL, REEVES, ROSS, THORNTHWAITE, 1998) afirma que se os dados extraídos estão armazenados em formato relacional, faz sentido o emprego de um framework relacional na implementação. Mas se não estão, não justifica a conversão destes dados para a utilização do framework porque o processamento a ser efetuado, basicamente, é de ordenação e sequencial.

3.2.1.2 Utilização de um gerador de código

Um gerador de código é um produto que obtém informações a partir de um repositório de metadados ou de especificações feitas através de ferramentas como o Rational Rose®, transformando-as em código fonte.

Os códigos, quando compilados e executados, processam o povoamento do DW. Eles são normalmente fontes C++, Java ou Cobol. Alguns geradores disponíveis no mercado são Prism Warehouse Executive, Chareton Passport e ETI Extract (WILLIAMS, 1997).

3.2.1.3 Emprego de uma ferramenta ETL

O emprego de uma ferramenta pode criar atalhos para a implementação de uma solução ETL. Existem várias delas no mercado entre as quais pode-se citar o DTS (*Data Transformation Service)* da Microsoft®, Sargent Solution da Sargent®, IBM® DataPropagator.

Embora sejam ferramentas que ajudem bastante o processo de povoamento dos DW, elas são geralmente dispendiosas e exigem configurações e treinamentos específicos, além do acompanhamento inicial de consultores. A grande vantagem delas é a rapidez com que os resultados são alcançados.

Mesmo sendo muito completas, nem sempre uma única ferramenta consegue atender a todas as tarefas definidas para o processo ETL. Quando isto acontece, é preciso utilizar um conjunto delas.

3.2.1.4 Adoção de uma solução híbrida

Em muitas situações, não é possível criar uma solução para o processo ETL através de apenas uma das alternativas apresentadas anteriormente. Na verdade, é preciso a combinação delas e cabe ao projetista, em concordância com o patrocinador do Data Warehouse, fazer a escolha. Eles devem analisar o problema, os objetivos, as expectativas dos futuros usuários, os recursos e estruturas disponíveis, entre outros, e tomar a melhor decisão.

3.2.2 Extração

A primeira operação executada durante o povoamento do DW é a extração cujo trabalho é obter os dados dos diversos sistemas transacionais e fontes externas. Estes dados serão processados na etapa seguinte: a transformação.

A extração dos dados se faz a partir dos sistemas fontes, bancos de dados, arquivos ou ainda de fontes externas que podem ser dos mais variados formatos (SINGH, 1997). Existem várias abordagens sobre a forma como a extração deve ser executada. Além disto, alguns princípios podem ser definidos para garantir o seu sucesso (GOODYEAR, RYAN, SARGENT, 1999):

- *Os dados devem ser precisos:* quando existem várias bases de dados contendo as mesmas informações, deve ser utilizada a mais precisa;
- *Os dados devem ser os mais recentes:* sempre que forem extraídos os dados das fontes, eles devem ser os mais atualizados;
- *Os dados devem ser fechados:* dados transitórios ou que ainda não estão completamente definidos fornecem conteúdo de má qualidade para o povoamento do DW. Por isto precisa-se sempre extrair dados fechados. Por exemplo, se o sistema financeiro de uma organização trabalha por mês de competência, deve-se sempre extrair os dados após o encerramento mensal.
- *Os dados devem ser completos:* a extração dos dados das fontes não pode ser parcial. Por exemplo, se os dados estão em duas bases distintas, a etapa de extração deve buscar o conteúdo em ambas. Além disto, se os dados são gerados em períodos diferentes, deve ser criado mecanismos que armazene o conteúdo, de forma incremental, até que ele esteja completo.
- Os dados devem estar o mais próximo possível de sua origem: quando os dados são obtidos através de diversas fontes auxiliares, via longas e/ou múltiplas rotas, sofrendo inúmeras transformações ao longo deste processo, até estarem disponíveis, podem perder acuracidade. Extrair dados diretamente das fontes principais, sempre que possível, é o melhor caminho.

Na criação do processo de extração, precisa-se definir o seu escopo, identificar e analisar as fontes de dados que serão utilizadas e especificar os programas e produtos que serão adotados na operação.

Identificar as pessoas envolvidas direta ou indiretamente com as fontes de onde serão extraídos os dados é uma ação importante, pois elas são fundamentais para o entendimento destes sistemas.

Quando não se entende que dados são importantes para o DW não se pode definir o que deve ser extraído das fontes. Por isto a escolha dos dados deve ser feita criteriosamente e baseada nas definições das informações que se espera obter do DW. Deve-se evitar extrair dados desnecessários que só aumentam o custo computacional e sobrecarregam os sistemas e hardware.

Os sistemas e produtos que serão desenvolvidos ou adotados na etapa de extração precisam ser direcionados às fontes de dados do DW. Eles devem ser

capazes de enxergar os dados, mesmo que estejam em formatos, sistemas operacionais ou locais diferentes e obtê-los, preparando-os para a fase de transformação.

Um desafio da fase de extração é tratar a heterogeneidade dos sistemas transacionais, que normalmente são criados em épocas diferentes e baseados em tecnologias diversas, fornecendo dados em muitos formatos. Além disto, alguns sistemas guardam o significado de seus dados no próprio código e não fornecem documentação esclarecedora, cabendo aos analistas descobrirem uma forma de entendê-los.

A falta de um padrão para a designação dos dados nas fontes é outro obstáculo ao processo de extração, pois dificulta o entendimento e localização do conteúdo que deverá ser extraído, gerando problemas do tipo:

- Elementos com a mesma designação, armazenados em bases distintas, tendo conotações diferentes;
- Elementos com designações diferentes, armazenados na mesma base, tendo a mesma conotação;

Todos esses fatores dificultam a compreensão dos dados a serem obtidos no processo de extração, fazendo com que entendimento deles fique dependente do conhecimento dos técnicos e profissionais que trabalham na organização com os sistemas fontes ou da documentação que houver.

Identificar o que será extraído é um trabalho que precisa ser feito por especialistas de negócio com a participação de gerentes de informática. Uma boa técnica para realizar esta tarefa é mapear os dados operacionais para os atributos das tabelas de dimensões e de fatos, e desta forma se definir quais desses dados serão transformados e integrados e quais serão descartados. Normalmente dados puramente operacionais como flags, status, não servem para o Data Warehouse.

Os meios utilizados para a extração dependem da forma de captura dos dados nas bases fontes. Podem-se usar aplicativos desenvolvidos na própria organização, em linguagens de programação como Cobol, Java, C++ ou outras. Se as bases de dados origens são relacionais, scripts na linguagem dos próprios SGBDs, como PL/SQL da Oracle®, podem ser mais convenientes. A última opção é a utilização de ferramentas de extração que são vendidas em pacotes. Elas permitem que o programador ou o usuário técnico obtenha dados de muitas origens de forma eficiente e rápida.

Os dados extraídos podem ser armazenados numa área de preparação (*staging* area) ou numa área de trabalho. A área de preparação geralmente reside no mesmo servidor de dados do DW e guardam informações em tabelas permanentes, cujos dados são transformados e transportados para o Data Warehouse. A área de trabalho é temporária e pode ser criada na memória ou em tabelas temporárias.

Há várias técnicas para se capturar os dados que serão utilizados no povoamento do DW e podem ser agrupadas em estáticas e incrementais (BOKUN, TAGLIENTI, 2000). A escolha da mais adequada é uma decisão tomada levando em consideração os requisitos do sistema de DW, as fontes de dados e os sistemas que os controlam.

3.2.2.1 Captura estática dos dados

A captura estática se baseia na aquisição periódica dos dados dos sistemas transacionais e fontes externas. A partir de agora, todos os sistemas que geram dados que são utilizados no processo ETL, tais como transacionais ou geradores de fontes externas, serão generalizados como sistemas fontes.

Esta técnica trabalha com a obtenção de dados em um determinado momento, sem se preocupar com as modificações que eles sofreram no período entre as aquisições. Por isto, não são implementadas rotinas que fiquem monitorando as alterações sofridas pelas bases operacionais.

O intervalo de tempo entre duas capturas deve ser suficientemente pequeno para gerar as informações desejadas pelos usuários do DW. Por exemplo, se os gerentes precisam consultar o faturamento mensal da organização, o intervalo entre duas capturas deve ser igual ou inferior a um mês.

A técnica de captura estática de dados não consegue registrar as modificações que ocorrem nos dados entre duas capturas. Mesmo assim ela pode ser considerada uma boa alternativa, pois evita sobrecarga dos sistemas fontes que não perdem performance tentando gravar continuamente as alterações que ocorrem em seus dados. Além disto, as capturas podem ser realizadas em horários apropriados, fora do “pico” de trabalho, quando os sistemas fontes estão mais disponíveis e ociosos.

Pode-se apresentar três técnicas derivadas da captura estática: *snapshot* dos dados, *timestamp* e captura baseada na comparação de arquivos sucessivos.

3.2.2.1.1 Snapshot dos dados

Esta técnica faz periodicamente a captura de dados dos sistemas fontes para a atualização do DW. Como as bases são capturadas completas, o processo torna-se muito pesado, exigindo muitos recursos computacionais.

Os dados extraídos, depois de tratados, substituirão completamente ou parcialmente o conteúdo da base do Data Warehouse. Quando a substituição é total, os dados anteriores do DW são eliminados através de comandos básicos como DELETE ou DROP.

3.2.2.1.2 Timestamp dos dados

Uma técnica mais eficiente, também mais difícil de ser implementada, é a *timestamp*. Através dela, somente dados que sofreram alguma modificação depois da última captura serão extraídos das bases dos sistemas fontes.

O problema desta técnica é que os sistemas fontes precisam registrar a data e hora quando os dados são modificados, criados ou eliminados. Normalmente isto é feito incluindo campos do tipo “datahora” nas tabelas e alterando os aplicativos para que gravem os momentos quando alterarem os registros.

Os campos “datahora” são utilizados no processo de extração para filtrar o conteúdo capturado, diminuindo o volume de dados transportados e aliviando a carga do DW.

3.2.2.1.3 Comparação de dados

Também conhecida como snapshot diferencial, esta técnica trabalha com o sistema de comparação de imagens sucessivas de dados adquiridos. Ela mantém duas imagens de dados referentes à última e penúltima captura dos dados dos sistemas fontes. O processo de identificação das mudanças é feito através da comparação destas duas imagens. Normalmente se usam as chaves dos registros e índices primários na tarefa.

Esta técnica é muito empregada quando não se pode alterar os sistemas fontes para que eles capturem as mudanças ocorridas nos seus dados. Além disto, o snapshot diferencial, assim como a técnica *timestamp*, gera uma quantidade de dados

menor para ser tratada e carregada no DW, embora a extração trabalhe com um grande volume de dados.

3.2.2.2 Captura incremental dos dados

A captura incremental é a técnica de extração onde se trabalha apenas com as alterações ocorridas nas bases de dados dos sistemas fontes. O processo ETL não utiliza toda a base de dados, mas apenas as mudanças ocorridas nos seus registros. A execução desta técnica é feita através do monitoramento das alterações ocorridas nos dados modificados pelos sistemas fontes. Cada inserção, alteração ou exclusão é registrada em um arquivo que posteriormente é usado pelo processo ETL para povoar o DW.

Há várias formas utilizadas para registrar as mudanças ocorridas nas bases fontes. Algumas delas se baseiam em características disponíveis nos SGBDs como gatilhos (*triggers)*. Outras, mais trabalhosas e restritivas, são realizadas através de intervenção nos códigos fontes dos aplicativos dos sistemas fontes.

As formas de implementação da captura incremental podem variar. Em algumas delas ocorre um overhead no processamento dos aplicativos e SGBDs porque estes sistemas precisam, além de suas funcionalidades normais, criar uma espécie de histórico das mudanças que realizam nos dados.

Nem sempre a captura incremental pode ser implementada. Quando não se tem acesso aos códigos dos sistemas fontes ou permissões para manipular as estruturas necessárias dos SGBDs fica impossível usar esta técnica para extrair os dados. Isto torna a captura incremental totalmente dependente dos sistemas fontes.

A captura incremental pode ser dividida em: processamento de gatilhos, extração de *log* e modificação de aplicação.

3.2.2.2.1 Processamento de gatilhos

*G*atilhos (*triggers*) são funções armazenadas nos SGBD que são disparadas sempre que um determinado evento acontece. Para que um gatilho seja executado, ele deve estar associado a um evento no banco de dados como: inserção, alteração ou exclusão de registros de uma tabela, criação de um índice.

A técnica de *processamento de gatilhos* consiste em utilizar os gatilhos dos SGBDs para registrar cada mudança ocorrida nas fontes de dados usadas no povoamento do DW. A ideia é criar gatilhos que registram em tabelas temporárias todas as alterações ocorridas nas bases dos sistemas fontes. Os dados destas tabelas temporárias, posteriormente são utilizados na alimentação da base do DW.

3.2.2.2.2 Extração do arquivo de "log"

A maioria dos SGBDs geram um arquivo, chamado de "log", contendo as alterações realizadas nas suas bases de dados. O princípio desta técnica, extração do arquivo de "log", é usar este arquivo, criado pelos SGBDs dos sistemas fontes, para extrair as mudanças ocorridas nos dados e usá-las no processo ETL do DW.

A maior vantagem da extração do arquivo de "log" é que não é necessário modificar programas dos sistemas fontes ou criar gatilhos (*triggers)* nos SGBDs para capturar as alterações nas bases de dados origem usadas no processo ETL.

Em contrapartida, existem várias desvantagens desta técnica:

- *Problemas referentes ao formato dos arquivos de "log":* como cada SGBD mantém um formato proprietário para os seus arquivos de "log", os programadores que estão implementando o processo de extração precisam entender este formato e criar aplicativos específicos para poder lê-los. Qualquer alteração no formato do arquivo de "log" implica em atualização das aplicações de extração.
- *Acesso ao arquivo de "log":* Os aplicativos que lêem os arquivos de "log" precisam ter acesso a eles, sem restrições. Nem sempre isto é possível.
- *Disponibilidade do arquivo de "log":* O arquivo de "log" deve estar disponível para os aplicativos usados no processo de extração. Se por qualquer motivo o arquivo de "log" for eliminado antes de ser obtido ou danificado ficando indisponível, o processo ETL falhará.

3.2.2.2.3 Modificação da aplicação

Esta é a última técnica de captura incremental de dados e consiste em alterar os sistemas fontes para que eles registrem as mudanças ocorridas nas bases de dados com as quais trabalha. É uma técnica bastante limitada, pois ela precisa de

intervenção nos códigos dos aplicativos, o que nem sempre é possível ou viável. Além disto, ela gera um grande overhead nos sistemas fontes que precisam registrar em bases temporárias cada alteração que faz nas suas bases de dados.

Como na técnica de captura baseada em gatilhos (*triggers)*, os dados registrados nas tabelas temporárias posteriormente são usados no processo de transformação e alimentação da base do DW.

3.2.3 Transformação

Os dados do DW são obtidos de diversas bases diferentes. Devido a este fato, é normal que muitas inconsistências e diversidade de formatos sejam encontrados nestes dados, o que inviabiliza a utilização deles sem antes serem tratados. O processo de transformação, etapa posterior à extração, prepara os dados de maneira que eles fiquem em condições de serem carregados no DW. A transformação dos dados ocorre ao nível de registro ou ao nível de atributo e as definições, ou características da transformação são armazenadas como metadados (BALLARD e HERREMAN, 1998).

Dependendo da natureza dos sistemas fontes, os dados obtidos de suas bases podem exigir transformações simples ou complexas, em uma única etapa ou em várias. O objetivo principal da fase de transformação é preparar os dados para que possam ser carregados no DW, e para isto precisam deixá-los em um formato adequado para esta operação. Como o modelo mais usado na construção das bases do DW é o dimensional, normalmente os dados transformados estão no formato ideal para este modelo.

É necessário que exista um conjunto de metadados que descrevam o processo de transformação. Ele será responsável por documentar e guiar as operações realizadas sobre os dados. Com metadados bem definidos, em uma linguagem que possa ser utilizada pelos técnicos e por ferramentas de apoio, o processo poderá ser compreendido e melhorado ao longo do tempo. Os metadados são, portanto, estruturas que guiam todo o processo ETL e funcionam como documentação do processo (TRONCHIN, 1996)

Os dados provenientes de sistemas heterogêneos precisam ser ajustados e integrados. Segundo INMON (1992), dados migrados para o DW sem estar integrados, não podem ser empregados no suporte a uma visão corporativa dos

dados. Mas para que a integração ocorra, inconsistências precisam ser eliminadas. Algumas delas, bastante comuns, podem ser identificadas:

- *Designação diferente para atributos com o mesmo significado*: campos como sexo podem ser representados como “M” e “F” ou “1” e “2”. Eles precisam ser ajustados para um domínio comum antes da base ser carregada no DW.
- *Formatos diferentes de campos para um mesmo dado*: datas, por exemplo, podem estar no formato dia/mês/ano ou ano/mês/dia, dentre outras. Embora todos representem datas, a diferença de formato pode invalidar as consultas do DW.
- *Atributos identificadores dos dados diferentes entre os sistemas fontes*: quando se trabalha com vários sistemas, pode haver a adoção de diferentes identificadores para uma mesma entidade. Por exemplo, “Usuário” pode ser identificado pelo seu CPF em um sistema e através de sua matrícula da organização em outro. É uma inconsistência que precisa ser corrigida com a adoção de um único identificador.
- *Conteúdo dos campos textos sujos, com excesso de espaços brancos ou abreviações variadas para a mesma palavra*: muitas consultas no DW podem falhar por causa deste tipo de problema. Textos com muitos espaços brancos, com abreviaturas erradas, ou com palavras escritas incorretamente. Geralmente a falha ocorre porque estes campos, nos sistemas transacionais, são preenchidos por pessoas que pela natureza humana cometem erros. Para que as inconsistências deste tipo sejam sanadas, precisa-se empregar algoritmos baseados em estatística, comparação de padrões.
- *Valores inválidos dos dados*: YOUNESS (2000) afirma que a validação dos dados aumenta a integridade do DW e a confiança na informação extraída. Dados inválidos prejudicam as consultas fornecendo informações que não servirão aos gerentes das organizações. Portanto, dados incorretos precisam ser corrigidos ou mapeados para representações válidas.

O processo de transformação age sobre os dados extraídos das bases fontes tratando-os e integrando-os de maneira que eles fiquem no formato ideal para povoar o DW. Algumas transformações comuns presentes no processo ETL são:

- *Integração*: esta operação obtém os dados dos sistemas fontes, combinando-os e mapeando-os para o modelo de dados do DW, gerando um conteúdo no formato específico, preparado para povoar o DW.
- *Limpeza e Validação*: como é utópico achar que os dados vindos dos sistemas do ambiente operacional são perfeitos e consistentes, e que não precisam ser validados, todo um processo de limpeza e tratamento dos dados precisa ser realizado para que eles fiquem consistentes e válidos.
- *Derivação e Sumarização*: Os dados vindos dos sistemas fontes possuem um baixo nível de granularidade, o que pode ser inadequado para o DW. Por isto precisam ser agregados. Além disto, algumas derivações podem ser geradas a partir dos dados originais, tais como criação de campos calculados e tabelas temporais.
- *Atualização de histórico*: Dependendo da forma como o processo ETL foi projetado, a atualização de dados já cadastrados na base do DW pode ser realizada. Independente da maneira como a operação é feita, deve-se ter em mente que relatórios e consultas gerenciais foram obtidos com base nos dados antigos do DW e alterar o valor deles gerará informações diferentes das que foram fornecidas aos usuários do sistema.
- *Ordenação*: a ordenação dos dados antes de serem inseridos no DW é uma atividade comum na transformação.

3.2.3.1 Transformações sobre os dados dos sistemas fontes

O modelo de dados normalmente utilizado no ambiente de DW é o dimensional, diferente do adotado no ambiente operacional. Obter dados dos sistemas fontes e adequá-los ao DW é uma operação que envolve muitas transformações. Primeiro porque os sistemas fontes não se preocupam com certas consistências que são importantes para a base do DW. Segundo porque o nível de qualidade dos dados destes sistemas é adequado as suas operações diárias, e visa atender aos seus requisitos funcionais que são diferentes dos requisitos do DW.

Cada sistema exige o nível de qualidade dos seus dados que atenda às suas necessidades. O DW precisa que sua base de dados atenda a certos requisitos não

funcionais ligados a qualidade para que possa ser útil, oferecendo informações confiáveis. A seguir estão alguns deles:

- *Consistência:* os dados do DW precisam ser consistentes, sem contradição. Duas consultas distintas, relativas a um mesmo assunto não podem apresentar resultados diferentes. Além disto, os somatórios nas pesquisas devem apresentar totais equivalentes à soma dos itens individualmente listados.
- *Unicidade:* O DW não pode ter valores diferentes para representar a mesma informação. Por exemplo, o DW não pode armazenar “J. dos Guararapes” e “Jaboatão Guararapes”, ambas representando a mesma cidade “Jaboatão dos Guararapes”;
- *Completude*: O DW não pode ter dados parciais, dentro do escopo definido, sobre os seus temas e assuntos. Se isto ocorrer, as informações obtidas do DW serão incompletas e serão inúteis para o processo decisório cuja execução é baseado nelas.
- *Precisão*: Os dados armazenados no DW precisam refletir a realidade apresentada pelas bases de dados dos sistemas fontes.

Serão apresentadas a seguir algumas transformações que são normalmente aplicadas aos dados operacionais para que estes possam ser incorporados ao DW, com a qualidade requerida. São operações visando a preparação do dado para o modelo dimensional adotado no ambiente de Data Warehouse.

3.2.3.1.1 Atributos com representação única

Muitas vezes a mesma informação está arquivada nas bases de dados operacionais em formatos, tipos ou unidades diferentes. É o caso dos estados da federação que podem ser registrados através de suas siglas ou nomes (ex: PE ou Pernambuco). Outro exemplo são os pesos dos produtos que podem estar armazenados em kg, toneladas ou gramas.

Durante o processo de transformação precisa ser escolhida uma única representação ou unidade para estes valores, convertendo-os quando necessário.

3.2.3.1.2 Adição de dados derivados

Entende-se por derivação durante o processo de transformação como a geração de novos dados a partir de outros existentes. Ela é realizada através da aplicação de fórmulas ou funções sobre os dados fontes, gerando novos valores que são usados na carga do DW. Geralmente o procedimento é feito ao nível de campos que são chamados de calculados.

Os campos calculados reduzem a quantidade de processamento e aumentam a velocidade das consultas do DW. Em contrapartida eles aumentam a área de armazenamento necessária à base de dados do DW. Um exemplo simples deste tipo de derivação é o campo “preço da venda” resultante da multiplicação do preço do produto pela quantidade vendida.

A quantidade de derivações realizadas a partir dos dados operacionais deve ser definida equilibradamente, pois um excesso delas pode fazer com que o DW cresça demasiadamente e se torne um problema.

3.2.3.1.3 Geração de novas chaves

As chaves identificadoras das dimensões do DW não devem ser as mesmas existentes nos sistemas fontes. As chaves criadas para as bases operacionais são projetadas para a união de tabelas normalizadas que sofrem transações diárias e atualização contínua de seus dados. O DW não pode usá-las como estão, pois, precisa atender as próprias necessidades demandadas pelo seu modelo de dados.

O DW precisa de chaves pequenas para suas dimensões, pois isto vai diminuir o tamanho de suas tabelas de fatos que ocupam muito espaço de armazenamento. Ele também tem que incorporar informações temporais às suas chaves.

3.2.3.1.4 Eliminação de dados que não são importantes

O processo de transformação precisa eliminar dados que não são relevantes para o DW. Tabelas, atributos, campos, tudo que não precisa constar nas bases de dados do DW deverá ser descartado.

Embora a operação de eliminação de dados desnecessários seja finalizada na transformação, esta atividade deve ser iniciada já no processo de extração que deve

selecionar apenas os dados dos sistemas fontes que são importantes para o DW. Isto diminui os recursos computacionais necessários ao processo ETL, aumentando o seu desempenho.

3.2.3.1.5 Agrupamento de valores em intervalos

A categorização de valores, mapeando-os para um conjunto de intervalos é um recurso que pode ser útil. A proposta é transformar valores contínuos, como idade das pessoas por exemplo, em faixas de valores (0-5 anos, 6-10 anos, ...). Esta transformação pode reduzir o tamanho das tabelas de dimensões e melhorar a clareza das informações referentes ao assunto. A desvantagem é o aumento do grau de granularidade com perda de informações no processo.

3.2.3.1.6 Tratamento para campos textos

Os campos textos, como nome e endereços, são problemáticos para o processo ETL, pois eles apresentam valores diferentes para a mesma informação. Isto normalmente ocorre devido à falta de um padrão adotado na entrada de dados, nos sistemas fontes.

Os problemas mais comuns encontrados nos campos textos são:

- *Abreviaturas incorretas*: a sigla “sen.” significando senhor.
- *Abreviaturas não padronizadas*: “João Carneiro dos S.” e “J. C. dos Santos”, ambos se referindo a mesma pessoa.
- *Dados incorretos*: “Recifo” significando Recife;
- Excesso de espaços em branco: “Rua dos Pinheiros “;

Em qualquer um dos casos anteriores a mesma informação se apresenta de forma diferente. O processo de transformação precisa tentar identificar estas falhas nestes campos textuais e tentar corrigi-las, aplicando regras, algumas vezes fazendo uso de estatísticas, para mapear cada entrada incorreta para valores que sigam um padrão pré-estabelecido.

3.2.3.1.7 Criação da dimensão tempo

O DW armazena dados históricos que são obtidos nas consultas. Muitos destes dados estão ligados ao tempo, e fazer comparações temporais sobre eles é importante para o processo decisório da organização. Por exemplo, uma organização fabricante de automóveis pode desejar saber qual foi o aumento de produtividade em relação ao ano anterior.

O nível de granularidade adotado durante a criação da dimensão tempo vai depender das necessidades dos negócios. Algumas operações precisam de acompanhamento diários enquanto outras mensais, trimestrais. Alguns DW armazenam em suas dimensões as datas puras, referentes ao dia e derivadas delas como a semana, a quinzena, o mês, o trimestre.

3.2.3.1.8 Tabelas não normalizadas

No DW, as tabelas não precisam estar normalizadas. Na verdade, este fato pode até reduzir a performance das consultas que precisarão fazer muitas junções. O processo de transformação deve tirar as tabelas da forma normalizada, quando necessário, unindo-as. Esta junção deve ser feita com tabelas oriundas de uma mesma base de dados operacional, que é o mais recomendável. No entanto, nada impede, se todos os cuidados e análises forem feitos, de se juntar tabelas vindas de bases operacionais diferentes. Neste caso, o sistema deve tomar bastante cuidado para não gerar inconsistências que irão fornecer dados de má qualidade ao DW.

3.2.4 Carga

Carga é o processo de povoar uma preexistente base de dados de um DW com conteúdo extraído e transformado das bases operacionais. As bases dos DW são construídas pelas organizações baseadas nas suas análises específicas sobre as suas necessidades em termo de informação.

A carga é a última etapa do processo ETL e espera-se que a transformação tenha sido realizada corretamente para que os dados estejam em condições de serem transportados para o DW. Durante o processo de carga, as seguintes características são importantes e devem ser consideradas (GOODYEAR, RYAN, SARGENT, 1999):

- *O tempo de carga:* dependendo da forma como a carga será feita, o DW pode ficar indisponível por um período. Isto porque algumas arquiteturas não permitem acesso ao DW pelo usuário enquanto ele estiver sendo atualizado. Mas é importante lembrar que, ao contrário dos sistemas transacionais, o DW não precisa necessariamente permanecer disponível todo o tempo, embora seja desejável. De qualquer forma, se o DW precisa ficar off-line por um período, é bom realizar as atualizações em horários especiais, como a noite ou nos fins de semana, quando o número de usuários é, normalmente, muito pequeno. Para sistemas de DW que são atualizados sem sair do ar, o projetista deve levar em consideração que as atualizações consomem muito processamento de máquina e memória, fazendo com que o ambiente fique lento, às vezes quase indisponível. Por isto, a mesma recomendação é feita: a carga deve ser executada em horário quando há o menor nível de atividade dos funcionários da organização.
- *A frequência da carga:* a medida em que os sistemas fontes vão atualizando as suas bases de dados, as mudanças precisam ser transferidas para o DW. Mas com que frequência estas transferências precisam ser feitas? Alguns fatores devem ser considerados. Quanto maior for o número de cargas de dados no DW, mais pesado ou indisponível ele ficará. Em contrapartida, um número pequeno prejudica as consultas porque elas não serão feitas sobre dados atuais. O projetista do ambiente de DW precisa decidir a frequência pesando a velocidade com que os sistemas fontes são atualizados, quanto tempo o processo ETL leva e quantos recursos ele consome, e quais são as necessidades da organização em relação ao DW.
- *A disponibilidade de recursos:* A quantidade de recursos computacionais disponíveis determina a velocidade com que a carga de dados é feita no DW. Quando existem sistemas dedicados, memória suficiente e processadores rápidos, ela ocorre com velocidade e sem prejudicar a disponibilidade do ambiente de DW.

O processo de carga pode ser executado basicamente de duas formas:

- *Bulk Load*: nesta forma de atualização, todos os dados do DW sofrem atualização. Normalmente os dados antigos do DW são eliminados e os novos carregados. O processo é pesado e demorado já que toda a base é substituída.

- *SQL Load*: esta é a forma de atualização mais eficiente. Através dela, o DW é atualizado apenas com os dados que sofreram mudanças desde a última carga.

O processo de carga pode ser tão complexo quanto for necessário, dependendo da arquitetura, modelo de dados, forma de atualização, sistemas fontes. No entanto, pode-se criar um roteiro básico que compreende as seguintes etapas:

a) *Criar imagens dos registros*: nesta etapa os registros são preparados para a carga. Eles devem estar no formato apropriado para serem carregados no DW. As estruturas de dados devem ser compatíveis com as do Data Warehouse de forma que a atualização aconteça corretamente;

b) *Criar agregações e suas chaves*: os dados devem ser agregados antes da carga. A princípio, durante o processo de transformação, todas as agregações previstas foram feitas;

c) *Carregar registros*: nesta etapa, os dados são efetivamente transportados para a base de dados do DW. Para que o processo ocorra com mais velocidade é recomendável, se o SGBD permitir, que a atualização automática dos índices seja temporariamente desligada. Outro aspecto pertinente é a integridade referencial. Se ela foi definida no DW, precisa ser respeitada e as tabelas devem ser atualizadas na ordem correta;

d) *Tratar as falhas*: um processo de atualização pode gerar inúmeras falhas que devem ser tratadas adequadamente. A qualidade dos dados depende disto. Registros rejeitados, chaves duplicadas, área de armazenamento insuficiente, tudo deve ser corrigido antes das atualizações do DW estarem disponíveis para os usuários. Quanto melhor for o processo de extração e transformação, menos problemas ocorrerão na carga;

e) *Construir os índices*: se foi possível desabilitar a atualização dos índices durante as etapas anteriores, este é o momento de reabilitá-la. Normalmente eles são reconstruídos pelos SGBDs. Este processo irá consumir algum tempo e processamento, mas é preferível que ele ocorra neste ponto do que anteriormente, durante a transferência de dados das áreas temporária para a base do DW;

f) *Garantir a qualidade dos dados*: esta etapa é fundamental e pode ser realizada manualmente ou automaticamente. Nela, as informações contidas no DW são checadas e validadas para que haja a garantia de que estão

corretas e aptas a serem utilizadas pelos usuários. Se houver problemas detectados durante a checagem, o processo ETL deve ser parcialmente ou totalmente repetido.

g) *Notificar os usuários sobre as atualizações*: Se o processo ETL obteve sucesso, neste momento o usuário do DW pode ser notificado sobre os novos dados disponíveis. Esta notificação pode ser feita nos próprios sistemas de consulta ou explicitamente através de e-mails operacionais ou avisos automáticos em tela.

No próximo capítulo serão tratadas as ferramentas ETL e seus principais aspectos.

4 FERRAMENTAS ETL

4.1 INTRODUÇÃO

Este capítulo é dedicado as ferramentas ETLs, ressaltando os prós e contras da adoção delas no desenvolvimento dos processos de povoamento dos Data Warehouse. Ele descreve quais são as principais características destas ferramentas que precisam ser avaliadas quando são adquiridas. Além disto, são apresentadas algumas delas como o DTS da Microsoft®. O objetivo das informações que serão apresentadas a seguir é levar o leitor a refletir sobre a importância de se utilizar as ferramentas ETLs.

O processo ETL pode ser realizado através de aplicativos codificados pela própria organização ou através de ferramentas adquiridas no mercado. Em geral muitas organizações optam por uma solução híbrida de código e produto (COLLIN, 2001). No entanto, algumas delas estão descartando a possibilidade de se utilizar soluções ETL prontas por causa dos custos, optando por construir suas próprias soluções, através do desenvolvimento de aplicativos. Embora esta decisão pareça economicamente vantajosa, adquirir um produto com anos de mercado, testado e consolidado, pode ser mais seguro e trazer resultados mais imediatos. Além do mais, o investimento de quem adquire ferramentas ETL tende a se diluir ao longo do tempo. Isto não significa que desenvolver os aplicativos ETL para o DW seja inapropriado. Em muitos casos, pode ser mais sensato e econômico. Por exemplo, quando o tamanho do projeto do DW é pequeno o suficiente para não justificar a realização de investimento em ferramentas, ou quando um só produto não atende às necessidades do projeto e precisa ser adquirido um conjunto deles ou se o produto mais adequado encontrado no mercado está ainda aquém das necessidades do projeto.

Os produtos ETL extraem dados das bases origens, transformam e carregam o DW, eliminando ou reduzindo o tempo gasto em desenvolvimento. Além disto, eles liberam os projetistas e seus colaboradores para pensarem nas funcionalidades dos sistemas ao invés de estarem investindo na implementação de aplicativos. Eles também operam baseados em regras definidas pelos programadores, executando o processo ETL ou gerando programas e scripts que o façam.

Para a aquisição de uma ferramenta ETL é preciso analisar muitos aspectos, além de saber exatamente o que o projeto de DW deseja construir. Além disto, deve

existir uma avaliação realista dos recursos que ela dispõe, tanto em infraestrutura, como financeiro e pessoal, para que os limites estejam bem delimitados e a aquisição fique dentro da sua realidade.

As grandes soluções ETL existentes são completas em termos de funcionalidades, mas difíceis de serem utilizadas nos projetos devido às suas complexidades. São ferramentas que precisam de um entendimento profundo para efetivamente ser usadas. Algumas delas precisam do suporte do fornecedor ou representante que é realizado através de consultorias e treinamentos. Outras trazem uma vasta documentação forçando a equipe que irá utilizá-las a se prepararem exaustivamente antes de as colocarem em operação. Tudo isto desestimula o pessoal do projeto que precisa esperar muito para ver os primeiros resultados concretos provenientes da utilização das ferramentas. Outro fator relacionado às grandes soluções é o excesso de funções, a maioria delas desnecessárias ao projeto. Quando a organização opta por uma ferramenta ETL, ela espera obter o mínimo necessário à solução de seu problema. Quando ela obtém mais do que precisa, termina não utilizando os recursos extras. Embora isto não represente um problema aparentemente, o investimento para adquirir a ferramenta poderia ser reduzido caso ela fosse menor.

Uma escolha equivocada de uma ferramenta ETL pode determinar o fracasso de um projeto de DW. É uma decisão complexa, pois enquanto uma solução pronta provavelmente pode trazer performance e funcionalidades, o desenvolvimento na organização traz maior adequação dos aplicativos aos requisitos existentes. O responsável pelo projeto precisa avaliar o problema e decidir qual é a melhor opção, que trará mais benefícios para o projeto. O fato é que a tecnologia avança aceleradamente e o número de dados gerados nos ambientes transacionais tem crescido nas organizações que, para poder acompanhar este crescimento, têm implementado os processos ETL apoiados por ferramental especializado. Por isto, cada vez mais pacotes prontos têm sido adotados.

Na prática, muitos fatores influenciam os projetos de DW e a decisão de adquirir uma ferramenta ou desenvolver os aplicativos necessários para se executar o processo ETL varia caso a caso. Alguns autores optam abertamente pela aquisição de ferramental, mas outros argumentam contra. Este texto não pretende resolver este impasse, mas apresentar os benefícios oferecidos pelos produtos existentes no mercado e mostrar que aspectos devem ser avaliados quando se opta por adquiri-los.

No final, serão apresentadas algumas ferramentas existentes, salientando algumas de suas principais características.

4.2 ANÁLISE DA AQUISIÇÃO E DESENVOLVIMENTO DE UMA SOLUÇÃO ETL

Quando se decide por construir um DW ou estender um existente, é necessário se ponderar os fatores que influenciarão o seu projeto. A decisão de se obter uma solução ETL pronta no mercado determinará a forma como todo o processo de construção e manutenção do ambiente de DW acontecerá. Por isto, a organização deve fazer uma criteriosa análise sobre o problema e responder a duas perguntas antes de tomar uma decisão:

- Por que adquirir um produto e não o desenvolver?
- Se vai ser adquirido, qual o produto ou o pacote de ferramentas é mais adequado ao seu problema?

4.2.1 Por que adquirir um produto e não o desenvolver?

Para se decidir sobre a aquisição ou não de uma ferramenta, faz-se necessária a análise de seus benefícios para os projetos de construção de DW. Em geral, as ferramentas apresentam soluções prontas para a maioria das atividades básicas e liberam os profissionais para focar exclusivamente no problema a ser resolvido. Para se ter uma posição mais segura, no entanto, alguns aspectos fundamentais para o DW, mostrados a seguir, precisam ser considerados nesta análise (FAISAL, 2003) :

- *Gerenciamento de metadados:* realizar investimento em metadados ou “metaprocesso” evita que a organização precise tratar os seus sistemas atuais como legados históricos no futuro. A grande maioria das ferramentas ETL dá suporte ao gerenciamento de metadados, fornecendo repositórios centralizados contendo informações claras sobre todo o processo que realizam. Algumas delas conseguem, inclusive, importar, exportar ou trabalhar com metadados definidos em outros produtos ou em formato considerado padrão pelo mercado. Dificilmente implementações caseiras conseguem trabalhar com metadados como as soluções prontas.
- *Gerenciamento de mudanças:* Os frameworks utilizados nos processos ETL precisam ser bem gerenciados e seu desenvolvimento controlado. Isto

implica na existência de controles de revisão, versão, erros e controle de versão. Embora sistemas desenvolvidos na companhia, se bem elaborados, possam conter todas estas características, ferramentas encontradas no mercado as fornecem, em geral, de forma mais integrada.

- *Performance e escalabilidade*: os sistemas desenvolvidos nas organizações resolvem problemas de performance de seus aplicativos melhorando os códigos das funções mais críticas. Já os produtos adquiridos tratam a performance através de serviços e monitoramento, conseguindo fornecer ao usuário informações gerais sobre o funcionamento do produto e soluções direcionadas para os problemas. Quanto ao aspecto escalabilidade, as ferramentas comerciais suportam várias plataformas, hardware e sistemas operacionais, o que permite às organizações crescerem na velocidade que desejarem, substituindo equipamentos à medida que for necessário.
- *Robustez e confiabilidade:* produtos obtidos no mercado são testados quando estão em desenvolvimento e na prática, ao longo de vários anos. Como eles enfrentam todo tipo de situação enquanto estão sendo utilizados, eles corrigem parte de suas falhas através das listas de erros reportados pelos seus usuários. Isto faz com que estes produtos sejam altamente tolerantes à falhas e ofereçam maior capacidade de recuperação e reabilitação quando os problemas acontecem. Além disto, eles possuem tratamento padronizado e específico para erros. São características importantes e difíceis de serem obtidas dos aplicativos caseiros. Mesmo com o avanço da engenharia de software que está permitindo a criação de softwares mais robustos, testados e de qualidade, faltam a estes sistemas o teste de campo em ambientes diversificados e o amadurecimento dos pacotes comerciais.

Algumas vezes os produtos obtidos no mercado não conseguem resolver satisfatoriamente um problema ou atender a um determinado requisito do processo ETL. Normalmente isto ocorre em grandes projetos de DW, onde aparecem tarefas complexas, difíceis de serem implementadas. Nesta situação as aplicações desenvolvidas nas companhias são as mais indicadas e muitas vezes as únicas que conseguem apresentar a solução necessária. Defensores dos pacotes de software prontos sugerem, neste caso, o emprego de ferramentas ETL que suportem programação, como o DTS da Microsoft®.

Algumas ferramentas ETL trabalham diretamente com pacotes considerados padrões de mercado, como SAP® e PeoplesSoft®. Elas usam templates configuráveis para extrair os dados destes ambientes e povoar o Data Warehouse. Isto é interessante para as organizações que conseguem rapidamente pegar as informações dos seus sistemas transacionais sem complicação. Embora o desenvolvimento nas organizações também possibilite a interação com os mesmos ambientes, o custo desta atividade é maior e exige mais trabalho da equipe de desenvolvimento.

Quando se tenta responder o porquê de se comprar ou implementar os aplicativos que executarão as tarefas ETL de um Data Warehouse, a resposta está numa conjunção de fatores que precisam ser avaliados. Desenvolver é uma tarefa árdua e que envolve muitos riscos, mas que fornece a solução mais ajustada ao problema. Comprar oferece resultados imediatos e riscos menores, mas o investimento financeiro é geralmente alto e uma escolha errada de um ou mais produtos pode decretar o fim do projeto do DW. Além do mais, produtos comprados trazem muitas funcionalidades desnecessárias ao projeto que só aumentam e complicam entendimento e utilização deles pelos usuários.

Na próxima seção, partir-se-á do princípio que a organização optou por usar uma ferramenta pronta e agora precisa decidir qual é a mais adequada ao seu projeto de DW. Neste cenário são apresentadas as características que devem ser usadas na avaliação das ferramentas disponíveis e são formuladas perguntas que precisam ser respondidas para definir a melhor escolha.

4.2.2 Qual o produto ou o pacote de ferramentas que é mais adequado?

Uma vez definido que será adotada uma ou mais ferramentas do mercado na construção do projeto do DW, a organização precisa selecioná-las. Esta atividade deve ser feita através de critérios técnicos e devem ser avaliados todos os quesitos relevantes. Normalmente uma boa avaliação inicia através do levantamento dos prós e contras de cada produto, onde cada característica deles deve receber uma pontuação e um peso. Os produtos que melhores notas tiverem deverão ser os mais indicados para serem comprados pela organização.

Durante o processo de avaliação das ferramentas, a organização deve obter todas as respostas para suas dúvidas. Ela deve consultar os fornecedores ou seus representantes, agendando visitas e demonstrações. Se for possível, deve obter

demos ou documentações específicas. É importante que todo este processo ocorra de forma planejada. Por exemplo, apenas os profissionais que tenham conhecimento técnico e experiência no uso de tecnologia precisam participar do processo de avaliação. Além disto, as demonstrações devem ser realizadas em épocas próximas, de preferência agendadas em uma sequência lógica e apenas os produtos que atendam aos requisitos mínimos devem ser demonstrados, evitando uma série de apresentações desnecessárias. Isto fará com que a equipe técnica foque suas energias nos melhores produtos.

O contato com o fornecedor ou representante precisa ser realizado com objetividade. Eles serão importantes no futuro, quando a organização adquirir os seus produtos, pois fornecerão o apoio logístico e técnico utilizado durante a construção do DW.

Finalmente, quando a equipe responsável pela avaliação da ferramenta ETL está formada, deve-se iniciar o levantamento das opções existentes no mercado e quais são as características de cada uma delas. Nas próximas duas seções serão apresentadas listas de critérios que devem ser observados pelas organizações quando resolvem selecionar um produto que pretendem adquirir (STEVEN, 2001) (SWEIGER, MADSEN, LANGSTON, LOMBARD, 2002). Estes critérios são analisados sobre duas óticas: do Data Warehouse e da ferramenta

4.2.2.1 Avaliação da ferramenta sobre a ótica do Data Warehouse

Neste tipo de avaliação, levantam-se todas as questões ligadas ao projeto de DW que se deseja construir ou expandir e determina-se o nível de adequação das ferramentas com elas. Os produtos que apresentarem melhores soluções para o maior número de itens levantados serão os selecionados.

As questões gerais consideradas neste texto são agrupadas em cinco categorias (STEVEN, 2001): complexidade, concorrência, continuidade, custo e conformidade.

4.2.2.1.1 Complexidade

As questões ligadas a esta categoria identificam a complexidade do ambiente de DW que será construído ou expandido. São elas:

- *Fontes de dados:* quantas fontes distintas de dados serão usadas para povoar o DW? A ferramenta trabalha com todas elas?
- *Arquivos de dados:* Se houver acesso a arquivos durante o processo ETL, quais os seus tamanhos e formatos? A ferramenta suporta os formatos? Ela consegue manipular grandes massas de dados de forma eficiente?
- *Teste e Depuração:* que tipos de testes serão implementados no projeto durante o seu desenvolvimento? A ferramenta consegue criá-los e executá-los?
- *Junções:* na extração dos dados, serão criadas junções entre tabelas pertencentes a bases diferentes ou a tipos de bancos de dados diferentes, ou entre bases relacionais e arquivos de dados? A ferramenta consegue realizar estas junções?
- *SQL:* a ferramenta suporta o Ansi-SQL ou extensões dele? Ela é eficiente ao executar as sentenças SQL e consegue trabalhar um grande volume de dados nestas execuções sem sofrer com a carga?

4.2.2.1.2 Concorrência

A concorrência avalia o número de programadores, durante o desenvolvimento do DW que podem usar a ferramenta ETL simultaneamente e o número de processos concorrentes que a ferramenta pode suportar.

- *Número de programadores:* quantos programadores irão trabalhar com a ferramenta ETL simultaneamente no projeto do DW? Que política de licenças é utilizada pela ferramenta: número de usuários simultâneos, número de instalações do produto?
- *Controle de acesso:* o ambiente de desenvolvimento e produção do DW, referente ao processo ETL, deve possuir que nível de controle de acesso? A ferramenta suporta o nível especificado?
- *Controle de versão:* como o sistema de controle de versão será implementado no projeto do DW? A ferramenta precisa estar integrada ao sistema de controle de versão ou pode usar solução proprietária neste quesito? Quais as funcionalidades que a ferramenta deve fornecer para se

enquadrar nos requisitos do ambiente de DW relacionados ao controle de versão?

- *Processos:* qual será o número máximo de processos simultâneos durante a execução da etapa ETL? A ferramenta deve dar suporte a processamento concorrente?
- *Falhas:* como será a recuperação do sistema quando houver queda no processamento ETL? A ferramenta consegue se recuperar de falhas no processo sem intervenção humana? Como os erros são registrados e como a equipe conseguirá usar estes registros?
- *Multiprocessamento:* o processamento ETL precisa suportar *multithread*? Existem limites na quantidade de *threads*? A ferramenta suporta multithread, em concordância com as especificações definidas pelo projeto para este item?

4.2.2.1.3 Continuidade

Este grupo de questões envolve a capacidade do ambiente de DW de evoluir ao longo do tempo, escalabilidade, e a confiabilidade das ferramentas diante de situações inusitadas.

- *Crescimento:* qual a previsão de crescimento do DW ao longo do tempo? A ferramenta consegue acompanhar este crescimento de forma satisfatória?
- *Mudanças:* se o processo ETL for alterado de alguma forma, a ferramenta consegue facilmente se ajustar às alterações? Qual o nível de dificuldade quando se precisa alterar rotinas ou *jobs* na ferramenta?
- *Hardware e SO:* qual o hardware e sistemas operacionais que serão usados pelo ambiente de DW? A ferramenta trabalha nestes equipamentos e sistemas? Existem versões da ferramenta para outras plataformas de hardware e software?
- *Falhas:* qual será o grau de disponibilidade que o DW precisa ter? Se houver falhas na rede, servidor, ou disco, a ferramenta consegue se recuperar quando o sistema voltar ao normal?

- *Upgrade:* com que frequência se planeja realizar *upgrade* da ferramenta ETL? Quais são os requisitos não funcionais para o *upgrade*, tais como facilidade e velocidade de instalação?

4.2.2.1.4 Custo

O custo se refere ao investimento que será feito na criação do DW e aos valores usados na manutenção do ambiente quando ele estiver em produção.

- *Preço:* quantas licenças da ferramenta ETL o projeto irá precisar? O preço das licenças é calculado pelo número de estações, de usuários, ou outra forma? Existem pacotes especiais dependendo do número de licenças adquiridas? Existe algum custo adicional referente à aquisição de componentes de terceiros como plugins, drivers que serão necessários para se usar a ferramenta?
- *Upgrade*: qual a política de atualização de software a ser adotada para o projeto do DW? Quais serão os custos médios com atualização da ferramenta? Qual o período médio entre duas atualizações?
- *Administração:* que tipo de profissional será necessário para administrar o processo ETL e seu ferramental? Eles serão em que número e quanto tempo eles estarão dedicados à atividade? Qual o custo destes profissionais?
- *Treinamento*: qual será o nível técnico da equipe responsável por implementar e manter o processo ETL? A ferramenta empregada precisa de que tipo de treinamento para poder ser operada ou configurada? Quanto será consumido em treinamento, durante quanto tempo, com que frequência, em que locais e com que infraestrutura?
- *Suporte:* quanto será necessário empregar em suporte à ferramenta ETL? Serão necessários consultores permanentes, temporários, eventuais?
- *Pessoal:* quais são os profissionais necessários para implementar e manter o DW? Para a ferramenta ETL que profissionais serão necessários contratar?

4.2.2.1.5 Conformidade

A conformidade trata da forma como a ferramenta se comporta diante das limitações do ambiente de DW que será implementado ou estendido.

- *Plataforma adotada:* qual a plataforma de hardware e software que será ou está sendo usada pelo DW? A ferramenta é compatível com ela? Existem incompatibilidades entre a ferramenta e outras usadas no processo ETL, como backup, antivírus?
- *Segurança:* existem sistemas de segurança de rede e de servidores? A ferramenta é compatível com eles?
- *Acesso às fontes de dados:* quais são as fontes de dados e suas restrições de acesso? As fontes estão localizadas em sistemas antigos, em plataforma restrita, ou em mainframe? A ferramenta tem acesso a estas fontes sem problemas?
- *Metadados:* que tipos de metadados serão usados no ambiente de DW? A ferramenta suporta todos eles?
- *Documentação:* que tipo de documentação será criado ou existe no DW e com que nível de qualidade. A ferramenta se enquadra neste padrão e contém todos os documentos necessários ao seu entendimento e uso?

4.2.2.2 Avaliação sobre a ótica da ferramenta

Nesta análise são examinadas as principais características das ferramentas ETL que devem ser consideradas quando elas estiverem sendo avaliadas. O foco, neste caso, é a ferramenta. Suas características, qualidades, arquitetura, como elas são utilizadas no processo ETL e quais as funcionalidades que a transforma em um artefato útil e valoroso no processo de povoamento do DW.

Os critérios levantados nesta análise são baseados nos estudos apresentados por SWEIGER (SWEIGER, MADSEN, LANGSTON, LOMBARD, 2002) :

4.2.2.2.1 Arquitetura do produto

Existem diferentes arquiteturas adotadas pelas ferramentas ETL. Algumas se baseiam na geração de códigos em linguagens consideradas padrões de mercado como C++, Java ou até SQL estendido. Outras funcionam baseadas em roteiros e definições que são executadas no próprio ambiente da ferramenta. O projetista tem que decidir qual a arquitetura é a mais adequada ao DW que pretende construir ou estender. Os principais quesitos que devem ser analisados são:

- *Plataformas suportadas:* existem no mercado muitas ferramentas que suportam uma ou mais plataformas. Este quesito determina se o processo ETL poderá migrar durante a sua existência para máquinas e servidores diferentes, aumentando a sua capacidade de executar as suas tarefas, sem precisar ser modificado ou ajustado. Algumas ferramentas ETL como as fornecidas pela Microsoft® trabalham apenas sobre uma plataforma operacional, no caso o Windows.
- *Designer modular*: esta característica é atribuída aos produtos que são vendidos em vários módulos. Ferramentas comercializadas desta forma têm duas vantagens: permitem que a organização adquira os módulos aos poucos, sem precisar fazer um investimento inicial acima do que deseja, e permitem que os produtos sejam estendidos através do surgimento de novos componentes.
- *Recuperação de falhas:* por mais planejado e correto que seja o processo de povoamento de um DW, sempre ocorrerão erros. Quando isto acontecer, a ferramenta precisará reportá-los aos administradores do sistema e prover meios para corrigi-los.
- *Acesso aos dados:* O principal elemento trabalhado pela ferramenta ETL é o dado. Portanto tudo que se refere a ele é importante. A ferramenta deve fazer acesso a todos os SGBDs do ambiente operacional e trabalhar com todos os tipos de dados tais como: texto delimitado, planilhas. Ela deve conseguir extrair e trabalhar com os dados de todos os sistemas fontes, não importando os seus formatos ou as plataformas sobre as quais eles estejam, e deve ter acesso direto às bases do Data Warehouse.

- *Suporte a processamento paralelo:* este quesito está relacionado à capacidade da ferramenta de processar diferentes tarefas simultaneamente, ou de tirar proveito dos servidores com múltiplos processadores.
- *Diversos:* além das questões que não foram enquadradas nos itens anteriores, têm-se: ajuda on-line, ferramentas de acompanhamento de performance, monitores de atividade.

4.2.2.2.2 Extração e carga

As questões aqui levantadas são relacionadas ao processo de extração e carga de dados no DW. Elas estão ligadas aos sistemas fontes e ao Data Warehouse, as plataformas onde estão lotados e os tipos e formatos de dados que possuem. Os quesitos principais são:

- *Conectividade:* este quesito define como a ferramenta consegue se conectar às diversas bases de dados fontes e à base do próprio DW. Muitas ferramentas ou fazem conexão direta às bases usando drivers nativos fornecidos pelos SGBD ou através de tecnologia padronizada do mercado como OLE DB ou ODBC;
- *Suporte a BD relacional e não-relacional:* A grande maioria dos BD atuais é relacional e as ferramentas ETL costumam trabalhar com eles. No entanto, não há garantia de que a ferramenta suporte outros bancos de dados como os mais antigos VSAM e IMS ou os mais novos como XML, Objeto-relacional.
- *Tipos de dados:* todas as ferramentas costumam trabalhar com os tipos de dados mais comuns como inteiros, reais, data. No entanto, outros mais específicos, como “imagem”, nem sempre são suportados. Esta limitação pode trazer problemas para os programadores que terão que criar subterfúgios para poder manipular estes tipos especiais. Outro aspecto é a conversão de tipos. Uma boa ferramenta consegue converter tipos mais comuns sem problemas, como inteiro para real, data para texto, caractere ASCII para EBCDIC, e geralmente provêm funções para a transformação de tipos mais complexos.
- *Suporte ao esquema dimensional:* Como o modelo dimensional é o mais adotado pelos ambientes de DW, espera-se de uma ferramenta de extração

e principalmente de carga que ela consiga trabalhar com este modelo sem problemas, criando dimensões, tabelas de fatos, agregações.

- *Suporte a produto padrão no mercado:* pode ser um fator decisivo na avaliação de uma ferramenta ETL a sua capacidade de trabalhar com os pacotes conhecidos no mercado como SAP.
- *Diversos:* qualquer ferramenta ETL empregada no processo de extração e povoamento de DW precisa fornecer uma interface padrão para acesso às diversas fontes de dados, suportar *bulk load* e *sql load*, ...

4.2.2.2.3 Transformação

A transformação precisa ser realizada corretamente, pois dela depende a qualidade dos dados que irão povoar o DW. Ferramentas que executam este processo tratam e preparam os dados fontes, integrando-os. Os principais aspectos a serem avaliados são:

- *Definição de regras:* regras de transformação são um dos requisitos mais óbvios das ferramentas ETL. Elas regem o processo de tratamento de dados e determinam quando as tarefas são disparadas e em que ordem elas acontecem.
- *Funções:* as transformações precisam ser executadas por funções definidas pelo programador ou pré-concebidas na ferramenta ETL. Estas funções compreendem de simples operações matemáticas e estatísticas até complexos tratamentos de dados.
- *Trabalho com aplicativos externos:* Nem sempre uma ferramenta ETL possui todas as funcionalidades necessárias à execução do processo de transformação. Quando ela não consegue executar uma tarefa, ela precisa acionar outro aplicativo que a faça. A capacidade de chamar um software externo, dentro do processamento, é um requisito importante para um produto ETL.

4.2.2.2.4 Gerenciamento de metadados

Toda ferramenta ETL deve trabalhar com metadados, pois eles são repositórios de informações sobre as bases de dados fontes, a base do DW e as operações realizadas no processo ETL. Além do mais, um metadado é sinônimo de documentação (TRONCHIN, 1996). As informações contidas nele descrevem o processo de povoamento do DW e servem de referência para todos os profissionais de tecnologia da organização. Os principais requisitos desejados nas ferramentas ETL em relação a metadados são:

- *Extensibilidade:* o metadado deve ser extensível para que informações sobre o processo ETL, que não foram previstas pelo fabricante da ferramenta, possam ser adicionadas a ele. Além do mais, a extensão do metadados deve ser feita facilmente, de forma organizada e com regras bem definidas.
- *Formato aberto:* a ferramenta ETL deve manter os dados dos seus metadados armazenados em um formato aberto, documentado e acessível a outros aplicativos. Algumas ferramentas ETL usam bases relacionais, arquivos texto, e as mais modernas, XML.
- *Compartilhar:* toda ferramenta ETL precisa ter a habilidade de compartilhar os seus dados de seus metadados com outros aplicativos ETL de forma transparente e automatizada. Além disto, ela também deve ser capaz de trabalhar com metadados de outras ferramentas, principalmente se estes estiverem em formatos adotados como padrão no mercado.
- *Integração com ferramentas de modelagem:* como os metadados das ferramentas ETL são repositórios de informações sobre as bases de dados fontes, a base do DW e as operações realizadas no processo ETL, eles devem importar e exportar informações dos principais produtos de modelagem de dados e esquemas de dados existentes no mercado.

4.2.2.2.5 Ambiente de desenvolvimento

Durante a criação do projeto de DW, é no ambiente de desenvolvimento da ferramenta ETL onde os programadores irão gastar a maior parte do seu tempo. Por isto este ambiente deve ser altamente produtivo, contendo configurações

individualizadas e facilidades que favoreçam o trabalho dos usuários que programam o processo ETL, tais como:

- *Ferramentas integradas:* as ferramentas de apoio ao desenvolvimento do processo ETL precisam estar todas no mesmo ambiente e à mão. Elas devem ser facilmente acessadas pelos programadores que devem evitar mudar de ambiente durante a execução do seu trabalho.
- *Ambiente gráfico:* atualmente a maior parte das ferramentas ETL fornece uma interface amigável e gráfica para os seus usuários. Este requisito é fundamental para quem programa. Produtos mais antigos, que ainda não migraram para o mundo gráfico devem ser evitados.
- *Linha de comando:* embora o ambiente gráfico seja altamente produtivo para o desenvolvimento, a velha e conhecida “linha de comando” ainda é muito útil. Ela agiliza certas atividades, pois o programador rapidamente executa pequenos e simples comandos sem ter que navegar nos menus da ferramenta ETL.
- *Sessão:* a ferramenta ETL deve implementar o conceito de sessão, que compreende um conjunto de definições, programas e dados relativos a um processo ETL específico. Isto permite que os programadores trabalhem em diversos projetos ETL, cada um fechado em si, sem haver sobreposição indesejada deles.
- *Suporte a SQL:* a ferramenta deve trabalhar muito bem com SQL, fornecendo todo o apoio e facilidades para os usuários em relação a esta linguagem de acesso aos dados, tais como *Wizards*, editores de textos especiais.
- *Depuração:* o ambiente de desenvolvimento precisa prover um depurador, de preferência com todas as modernas opções disponíveis nas melhores linguagens de programação, tais como: acompanhamento passo a passo, *breakpoints*, *call stack*. Quanto melhor for o depurador da ferramenta ETL, mais produtivo será o trabalho dos programadores e menos erros ocorrerão no processo de povoamento dos DW.
- *Editor de dados*: uma boa ferramenta ETL precisa, como parte do seu ambiente de desenvolvimento, fornecer acesso aos dados, através de planilhas e formulários, permitindo que edições sejam realizadas.

- *Controle de versão e acesso:* como os programadores trabalharão em equipe, é indispensável que exista um controle de versão capaz de gerenciar as alterações realizadas nas definições do processo ETL. Além disto, o acesso às funcionalidades das ferramentas e bases de dados deve ser controlado, obrigando a ferramenta a possuir um bom controle de acesso.

4.2.2.2.6 Administração

Após a implementação do DW e entrada em produção, a administração dos seus dados é o conceito chave para o seu bom funcionamento. Não adianta o processo ETL funcionar perfeitamente se não houver um controle do que ele está fazendo e o que ele está gerando no DW. Por isto, o processo ETL precisa ser administrado através das suas ferramentas que devem ser avaliadas nos seguintes quesitos:

- *Administração centralizada:* é fundamental que todo o gerenciamento do processo ETL esteja centralizado em um único lugar, mesmo que os dados e *jobs* estejam armazenados e sendo executados, respectivamente, em plataformas ou equipamentos diferentes e distribuídos. Isto evitará o descontrole e facilitará a organização e o acompanhamento de todo o processo.
- *Correção de falhas:* quando algum erro acontece durante o processo ETL, a ferramenta deve apontar o problema e fornecer os meios para a correção ou reinicializarão de todo o processo. Além disto, através de mecanismos como alerta, e-mails de aviso, as notificações feitas pela ferramenta devem alcançar as pessoas responsáveis por corrigir as falhas.
- *Auditoria:* quando um processo ETL termina, o administrador do DW precisa saber quantos dados foram processados em cada operação, de onde os dados foram obtidos, como foram transformados e armazenados, o tempo que o processo levou, e onde aconteceram os maiores gargalos. Isto tudo faz parte de auditorias que a ferramenta ETL deve fornecer.
- *Scheduling:* o processo ETL tem muitas etapas que ocorrem em paralelo ou em sequência, a maioria com dependências entre si e disparadas conforme uma série de regras e eventos. Devido a esta complexidade, ferramentas

que possuem a capacidade de agendar, dentro de uma estrutura lógica, as tarefas que compõe o processo ETL são mais úteis e melhor avaliadas.

De forma geral, estes são os aspectos analisados das ferramentas ETL quando estão sendo avaliadas pelas organizações durante o processo de aquisição. A seguir será apresentado um estudo sobre as principais ferramentas e produtos disponíveis no mercado, ressaltando as suas principais características.

4.3 AVALIAÇÃO DE FERRAMENTAS ETL

Como foi visto nas seções anteriores, avaliar com cuidado a ferramenta que se deseja adquirir é importante para evitar o investimento em produtos que não atendam às necessidades do projeto de DW. Existem muitas opções disponíveis tais como DTS Microsoft®, DH ETL, DataStage e são organizadas por categoria. Algumas delas são vendidas em pacotes contendo vários produtos, cada um direcionado para uma etapa do processo ETL.

Nesta seção, serão apresentadas algumas ferramentas que foram consideradas importantes, ou porque tinham por trás uma grande organização garantindo-as, como a Microsoft®, ou porque havia muitas referências a respeito dela e, portanto, são padrões de mercado.

Nesta apresentação, o objetivo não é detalhar ao máximo cada um dos produtos escolhidos, mas mostrar quais são suas principais características, como são organizados, quais são as suas vantagens e problemas, como são montadas as suas arquiteturas, dando uma ideia global de seu ambiente e como eles resolvem o problema de povoamento de dados dos DW.

4.3.1 MS SQL Server

O MS SQL Server é o banco de dados cliente-servidor da Microsoft®. Este produto contém um conjunto de componentes que juntos fornecem tudo que é necessário à operação, gerência e manutenção de um BD. No seu pacote estão contidos: o banco de dados relacional, ferramentas destinadas à criação e manipulação de DW e MS English Query (MICROSOFT, 1998a). O ponto mais positivo do SQL Server é a integração alcançada por suas ferramentas, além da interface gráfica com opções que ajudam o usuário a ter controle sobre os dados e as

operações realizadas sobre eles. Estas facilidades sempre foram marcas registradas dos produtos da Microsoft®.

Para a criação de um DW, o SQL Server disponibiliza o Data Warehousing Framework, que provém uma arquitetura aberta para que o produto possa ser integrado com outros aplicativos, metadados integrados, serviços para gerenciamento e análise do banco, além de serviços destinados ao processo ETL como importadores e exportadores de dados, serviços para extração, transformação e limpeza de dados.

A Microsoft® desenvolveu o padrão tecnológico OLE DB, que fornece interoperabilidade entre múltiplas plataformas de forma que elas consigam ter acesso a diversas fontes de dados diferentes. Esta tecnologia é usada pelo SQL Server para ter acesso às fontes de dados externas, permitindo que seus processos ETL consigam extrair o conteúdo de quaisquer sistemas fontes.

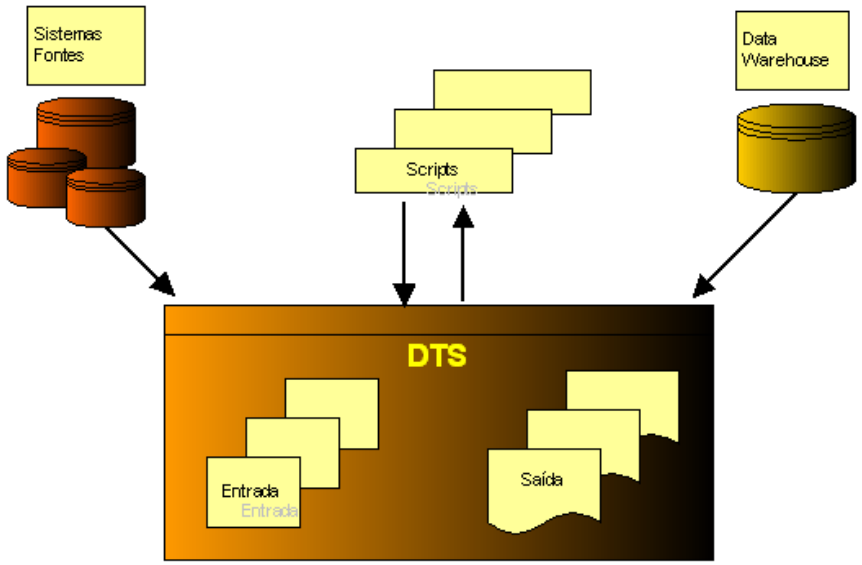
Outra característica da arquitetura do SQL Server, ligado a DW, é o repositório integrado de metadados. Ele armazena informações sobre todos os componentes e seus relacionamentos, permitindo que todos os serviços ligados ao processo ETL consigam obter, em um local compartilhado, informações sobre os dados.

O SQL Server fornece o *SQL Enterprise Manager*, uma ferramenta onde o programador pode criar tabelas, relacionamentos, *stored procedures*. Ela pode ser usada na criação das tabelas de fatos e dimensões do DW. Trata-se de uma ferramenta visual, onde todas as entidades do banco de dados ativo são apresentadas de forma organizada.

Para realizar o processo ETL, o SQL Server disponibiliza o DTS ou *Data Transformation Services*. Este produto permite que os programadores definam e executem todas as etapas envolvidas no processo de povoamento do DW, fornecendo serviços de importação, exportação, extração. Além disto, o DTS usa o repositório integrado de metadados do SQL Server para orientar as suas operações.

A figura 9 apresenta a arquitetura resumida do DTS.

Figura 9 - Arquitetura resumida do DTS



Fonte: o autor (2004)

O SQL Server, através de interfaces gráficas, permite que o programador consiga definir vários serviços no DTS interativamente, sem precisar codificar. Além disto, como a Microsoft® fornece integração de suas ferramentas e tecnologias, o programador também pode utilizar scripts ActiveX, construídos em VB Script para realizar tarefas ligadas ao processo ETL.

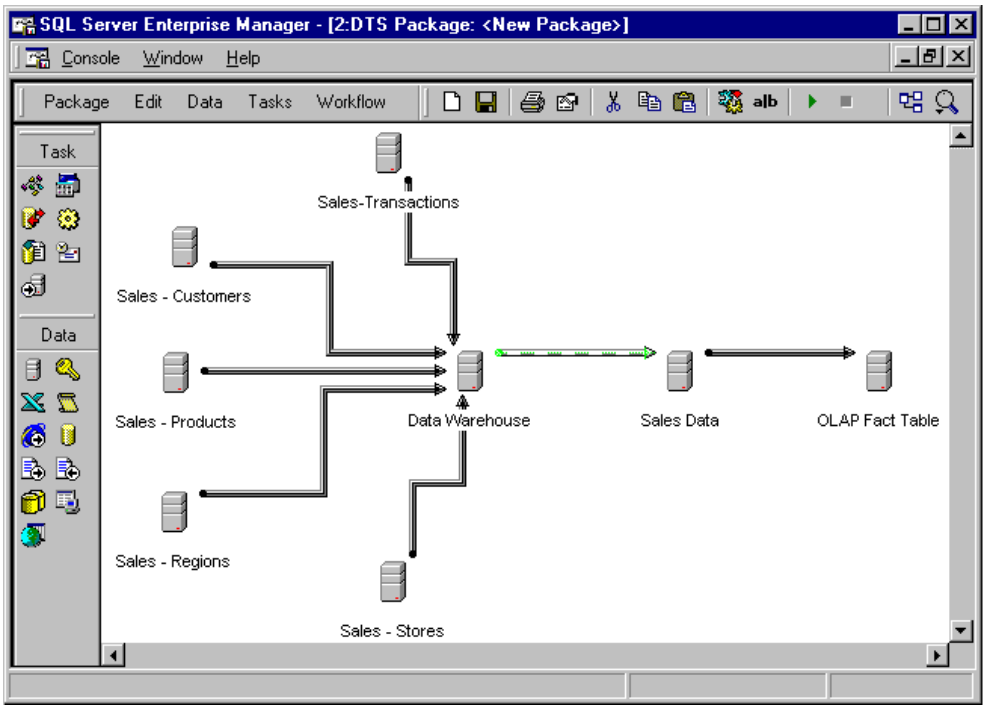
O DTS tem como principais características:

- Linguagem de desenvolvimento Transact-SQL, uma extensão do SQL.
- Integração completa com o repositório central de metadados, registrando nele cada operação sofrida pelos dados;
- O DTS package, que fornece entre outras funcionalidades a capacidade de definir um conjunto de tarefas e atividades, além da sequência como elas são executadas. O DTS package permite que o administrador do processo ETL agende as tarefas para que elas ocorram em uma dada ordem e em um determinado momento. A figura 10, obtida em (MICROSOFT, 1998b) apresenta uma visão desta ferramenta.

Embora o SQL Server ofereça um ambiente bastante completo para o desenvolvimento do processo ETL, a limitação imposta pela Microsoft® aos seus produtos que só podem ser executados sobre o seu sistema operacional é um obstáculo às organizações que desejam utilizá-lo e que consideram importante a independência de plataforma para os seus projetos.

Além disto, o produto é direcionado totalmente para o banco de dados da Microsoft®, impedindo o seu uso em outros ambientes de BD. Esta limitação impede a sua utilização por empresas que adotaram outros SGBDs na implementação de seus Data Warehouse.

Figura 10 - DTS package



Fonte: o autor (2004)

4.3.2 DataStage

O *DataStage* é um produto fornecido pela Ardent Software®. Ele é apresentado em três formatos: o *DataStage*, *DataStage Enterprise* e *DataStage XE*. Seu foco é a realização do processo ETL dos DW e pode ser usado para movimentar dados entre quaisquer fontes heterogêneas. O pacote inclui, além das ferramentas ETL, um gerenciador de qualidade de dados, o *Quality Manager*, e um gerenciador de metadados, antes chamado de *MetaStage* e atualmente conhecido como *Metadados Management*.

As principais características do *DataStage* são:

- O componente *MetaStage* do *DataStage XE* que é capaz de gerenciar metadados de forma eficiente e flexível.
- Fornecimento de um gerador de código Cobol, que oferece aos programadores desta linguagem a capacidade de construir programas

inteiros sem precisar escrever nenhuma linha de codificação, apenas através de definições realizadas no DataStage.

- Ambiente de desenvolvimento para implementação do processo ETL. O DataStage fornece a sua própria linguagem de programação e oferece facilidades como um depurador integrado.
- Uma interface fácil de ser usada, direcionada aos programadores que desenvolvem o processo ETL.
- O *Quality Manager*, uma ferramenta fornecida no pacote do produto que possibilita a realização de auditorias sobre os dados do DW. Esta ferramenta oferece uma série de funções pré-definidas para realização de testes, filtragens e a capacidade de executar qualquer sentença SQL.

O *DataStage* suporta a extração de dados de praticamente todas as fontes, incluindo aquelas que estão em mainframes ou pertencentes a pacotes de aplicações como SAP®. A lista a seguir mostra as fontes que são acessadas pelo *DataStage*: IBM® DB2, Informix® Dynamic Server, Informix® Red Brink, Microsoft® SQL Server, MQ Series, Named Pipes, Oracle®, Sybase® Adaptive Server, UniVerse, UniData, ADABAS®, VSAM/QSAM/ISAM, Teradata®, Flat Files, FTP. Além disto, o *DataStage* também pode acessar dados através de ODBC ou OLE DB, tecnologia criada pela Microsoft®.

O *DataStage* é um produto que trabalha em muitas plataformas, o que lhe confere alta portabilidade. As plataformas suportadas por ele são: Windows NT, Compaq Tru64, HP-UX, IBM® AIX, Sun® Solaris, OS/390 e Linux.

O ambiente de desenvolvimento do *DataStage* oferece centenas de funções de transformação, das mais simples as mais complexas. Além disto ele é completo, com linguagem embutida de script e depurador. Outra facilidade deste ambiente é a integração com rotinas externas escritas em C, além de suporte a *ActiveX*. O *DataStage* também dá suporte aos programadores Cobol. Ele é capaz de gerar códigos nesta linguagem a partir das especificações realizadas dentro do seu ambiente.

O *XML Pack*, pacote incluído no produto, oferece os recursos necessários para que se possa trabalhar com XML facilmente. Ele permite que se obtenha ou forneça informações usando o formato XML, podendo ser trocado dados ou metadados através deste padrão. Além disto, o XML Pack consegue gerar dinamicamente a DTD que descreve a estrutura dos documentos XML.

O *DataStage* fornece no seu pacote XE o *MetaStage*, um gerenciador completo de metadados, que oferece amplo suporte ao compartilhamento de metadados, importante característica de um DW, que dá ao usuário compreensão exata das estruturas e complexidades de seus dados.

O *DataStage* oferece um excelente desempenho pois trabalha com tarefas paralelas e consegue realizar manuseio de dados na memória. Ele trabalha completamente integrado com pacotes de aplicações consenso no mercado como o SAP.

Finalmente, o *DataStage* possui *Quality Manager*, um aplicativo que permite realizar auditoria, monitorar e gerenciar a qualidade dos dados do ambiente de DW, fornecendo aos usuários um quadro completo sobre os sistemas fontes e o Data Warehouse.

Um ponto negativo para o DataStage são problemas ligados a integração de alguns dos seus subprodutos, como o Quality Manager. Este fato é justificado porque estes subprodutos foram incorporados ao pacote durante a aquisição de empresas como a Prism® pela Ardents® e ainda não houve tempo hábil para uma completa integração dele com os demais módulos.

Sem dúvida, o DataStage é um dos produtos mais completos atualmente disponível no mercado. Existem representantes do produto em praticamente todos os pontos do mundo como Japão, Europa, Ásia, Américas e até na África.

4.3.3 DataHabitat – ETL

O DataHabitat ETL ou simplesmente *DH ETL* é um produto licenciado pela DataHabitat® Corporation e faz parte de um pacote composto por mais outros dois: o *DH Query* e o *DH Web*. Este produto gera códigos para o DTS da Microsoft® baseado em metadados definidos pelos usuários através de sua interface gráfica.

O DH ETL permite que o usuário interativamente defina transformações, funções, sumários, tabelas, condições lógicas, filtros, resumindo, todas as operações e entidades ligadas ao processamento ETL. As definições são armazenadas em repositório de metadados e são utilizadas na geração dos códigos DTS.

O grande benefício do DH ETL é o fornecimento de uma visão gráfica do processo ETL, escondendo do usuário as complexidades do Microsoft® DTS. Em contrapartida, este produto é totalmente direcionado para o SQL Server da Microsoft®

e isto limita o seu campo de atuação e o transforma em um ambiente limitado a uma plataforma especializada.

As principais características do DH ETL são:

- Construção de pacotes DTS sem programação
- Interface direcionada a metadados
- Suporte a fórmulas com funções e condições
- Construção de métricas e indicadores de performance
- Capaz de criar esquemas de BD baseados no modelo estrela, floco de neve
- Criação de documentação automática do Data Warehouse.

O DH ETL é uma ferramenta útil para programadores do SQL Server DTS que têm a produtividade aumentada e o trabalho facilitado através do emprego das funcionalidades que ela oferece.

4.3.4 DecisionPoint

DecisionPoint pertence a uma família de pacotes destinados a prover soluções para Data Warehouse. Ele cria dimensões, índices, domínios, relatórios, entre outros artefatos pertencentes aos DWs, além de trabalhar com os mais conhecidos ERP do mercado tais como SAP® e PeopleSoft®. Sua interface gráfica é estendida aos seus aplicativos destinados à administração dos DW. Além disto, esta ferramenta disponibiliza um gerador de relatórios que pode ser utilizado por usuários com pouca experiência em banco de dados.

Uma característica que valoriza a ferramenta é sua natureza de pré-empacotamento. Ela fornece pacotes prontos para uma série de domínios de negócio que vai da área financeira até a humana. Isto permite, segundo a empresa fabricante, que se implemente um DW em poucas semanas, com baixo custo.

As características mais relevantes do DecisionPoint são:

- Preço fixo: o produto oferece uma política de preço e modelo de implantação e desenvolvimento de DW que elimina as variações de custos normalmente apresentadas por outras ferramentas de acordo com configurações e pacotes.
- Suporte nativo a vários domínios de negócio como financeiro, recursos humanos, produção e distribuição, análise de vendas, serviços. Cada

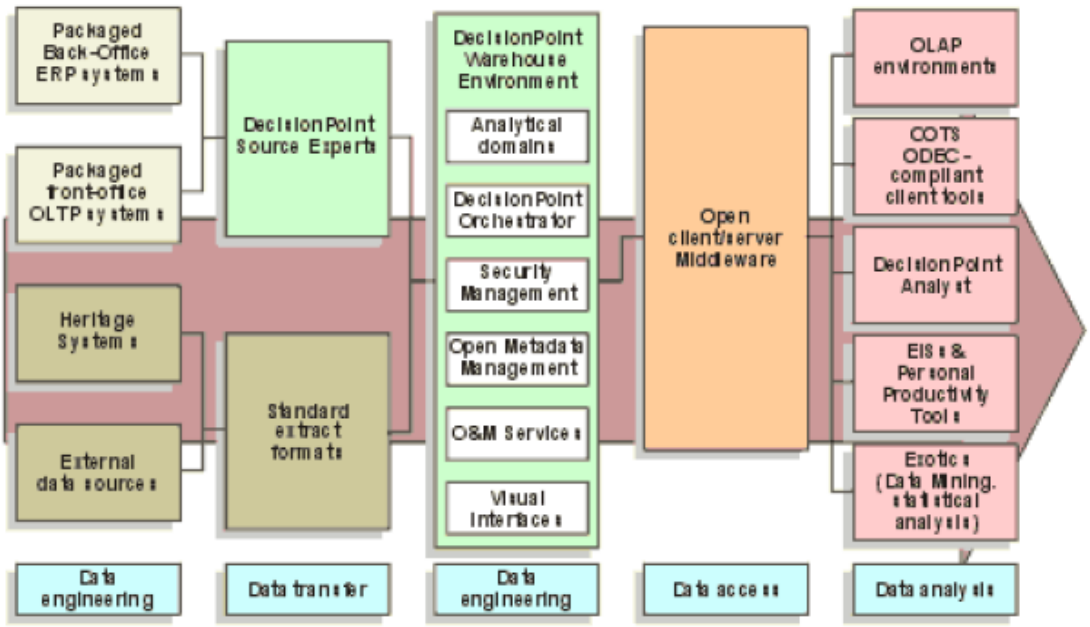
domínio é quebrado em subdomínios com uma série de detalhes que devem atender as necessidades dos projetos de DW.

- Um ponto negativo para o produto é que ele não suporta entrada de dados realizada diretamente pelo usuário.

A figura 11 mostra a arquitetura do DecisionPoint. Os principais elementos ilustrados na arquitetura são:

- *DecisionPoint Source Experts*: provedor de funcionalidades que aceleram a leitura de dados dos ERP com os quais o DecisionPoint trabalha: SAP®, Oracle® e PeopleSoft®;
- *DecisionPoint Analytical Domains*: fornecedora de soluções para BI – business intelligence em áreas de negócio específicas;
- *DecisionPoint Analyst*: ferramenta front-end utilizada com o Analytical Domains. DecisionPoint Analystic é um gerador de relatório ad hoc que cria saídas no formato HTML ou Microsoft® Excel;
- *DecisionPoint Administrative*: usado na administração de metadados, agendamento de tarefas, gerenciamento de segurança e operações de manutenção de tabelas do esquema dimensional do DW.

Figura 11 - Arquitetura do DecisionPoint



Fonte: o autor (2004)

4.3.5 ETI (Evolutionary Technologies International) – Extract

ETI é uma ferramenta ETL simples se comparada com outros grandes produtos, mas eficiente no transporte de dados entre os sistemas fontes e o DW. Ela move grandes volumes de dados com velocidade e eficiência.

A proposta da ferramenta ETI é gerar programas que irão realizar os processos ETL. Suas principais características são:

- Habilidade de processar grandes lotes de dados de forma rápida e eficiente.
- Flexibilidade na incorporação de complexas transformações;
- Gerenciamento de alto nível dos metadados que controlam os processos ETL;
- Geração de códigos que podem ser integrados com os sistemas de gerenciamento de infraestrutura.

Uma limitação da ferramenta ETI é seu acesso nativo restrito apenas a alguns tipos de bases de dados. No entanto, ela pode ser entendida facilmente e fazer acesso a outros tipos não nativos. Outra questão que pesa contra ela é a necessidade de um administrador experimentado para seu gerenciamento completo, o que implica no treinamento forte deste profissional encarregado de sua administração;

Enquanto a administração da ferramenta ETI exige um profissional altamente qualificado, a programação dos processos pode ser feita através de uma interface intuitiva que diminui a curva de aprendizado necessária à sua utilização.

ETI possui ótima reputação no mercado apesar do pequeno porte da empresa que a produz. A ferramenta trabalha com grande volume de dados e em complexos ambientes. Atualmente, a fabricante tem se associado com grandes organizações tal como a IBM para dar maior credibilidade ao seu produto e aumentar a sua interoperabilidade com as soluções consagradas no mercado.

4.3.6 PowerCenter 5

O PowerCenter é uma família de produtos que movimenta e integra dados. Ele trabalha com sistemas a nível departamental, criando e gerenciando Data Marts até a nível organizacional como Data Warehouse distribuídos. Seu fabricante, a “Informática”, foi fundada em 1993 como fornecedora de serviços, especializada em migrar dados para o ambiente client/Server. Com a aquisição da Influence Software,

uma companhia focada em aplicações direcionadas para a análise de dados, e Zimba, especializada em técnicas de personalização, seu foco mudou e o PowerCenter foi criado como principal produto.

O PowerCenter 5 é composto pelos seguintes subprodutos:

- *PowerCentre5*: este subproduto é definido pelo fabricante como a solução para o desenvolvimento e gerenciamento de DW. Ele possui uma série de facilidades e suporte nativo a XML.
- *PowerCenter 5 Data Integration*: este produto é a extensão do PowerCenter voltado aos grandes projetos de DW, contemplando sistemas distribuídos em diversos servidores e plataformas. Ele prover escala e poder para soluções voltadas a Enterprise Data Warehouse.
- *PowerMart 5*:Ao contrário do subproduto anterior, este visa os projetos departamentais, provendo soluções para os Data Marts. Embora a "Informatica" esteja centrada em fornecer soluções para grandes projetos de DW, o PowerMart se coloca como um ponto de partida que provém a transição entre DM e DW.
- *PowerConnect*: este subproduto fornece conectividade com SAP/R3, PeopleSoft, Siebe e MQSeries. Na verdade, ele vai além, entendendo os metadados destes pacotes citados que são bem integrados com os do PowerCenter.
- *PowerPlug*: este subproduto importa e reusa os metadados de várias ferramentas de modelagem tais como ERWin, Oracle® Deisgner, Sybase PowerDesigner e Sybase S-Designer. Além disto, o produto vem com o módulo Metadata Exchange (MX2) API que provém suporte para outros ambientes.
- *PowerBridge*: este subproduto provê facilidades para exportar o metadados OLAP do repositório do PowerCenter para um formato que pode ser lido pelo servidor OLAP Hyperion Essbase que é entendido pelo servidor OLAP do DB2 e Showcase Strategy.

4.3.7 Considerações gerais

O número de ferramentas disponíveis é grande atualmente. Além das apresentadas neste texto, existem outras como Axio e a Kalido. As mais completas são multiplataforma, estando disponíveis em versões para os principais sistemas operacionais como Unix, Linux e Windows. Além disto, a disponibilidade de funcionalidades voltadas a WEB é uma tendência nova destas ferramentas.

Muitos produtos apresentados nasceram a partir de junções ou aquisições entre empresas de tecnologia que uniram os seus *knowhow* e construíram os seus pacotes. É o caso da ferramenta PowerCenter, cuja vendedora, a Informatica, adquiriu a Influence Software e a Zimba.

A diversidade de ferramentas é grande e a forma como trabalham e os aspectos que atendem no processo ETL variam. No entanto, elas apresentam características comuns. Por exemplo, todas trabalham com metadados que podem estar em formatos proprietários ou padrões de mercado como o Hyperion Essbase. Além disto, elas acessam a maioria dos grandes bancos de dados como Oracle®, DB2 e muitas possuem uma forte integração com uma variedade de ambientes ERP como o SAP/R3 e o PeopleSoft®.

5 O AMBIENTE HYDRUS

5.1 INTRODUÇÃO

Nos capítulos anteriores foram vistos os principais aspectos referentes ao ambiente de Data Warehouse, especificamente o processo ETL – *Extract Transformation, Load*. Foram apresentados os conceitos gerais sobre o DW, sua arquitetura, seus componentes e as principais características do seu desenvolvimento. Para o processo ETL foram vistas as fases, os problemas envolvidos no processo e alguns pontos de vista a respeito das soluções adotadas. Finalmente foi apresentada uma discussão sobre as ferramentas ETL, porque adotá-las e quando elas não são apropriadas ao ambiente de DW. Foram vistos também os requisitos a serem observados quando se decide usá-las.

Agora será visto o tema central desta dissertação que é o projeto Hydrus, um ambiente extensível de apoio à construção de ferramenta ETL.

5.2 HISTÓRICO

O projeto Hydrus nasceu em 2000 e seus primeiros conceitos foram aplicados a um problema real quando a Chesf, Companhia Hidrelétrica do São Francisco, resolveu criar o SCOPE, um Data Warehouse que armazenava informações sobre os custos da companhia, disponibilizando relatórios estratégicos sobre os recursos empregados nos diversos setores da empresa. O objetivo do SCOPE era fornecer à gerência informações sobre os recursos gastos, como estavam sendo aplicados, apresentando os dados sobre várias visões: pessoal, operacional e administrativo. O sistema, feito de forma quase artesanal, levou ao entendimento das dificuldades existentes no processo de criação de um DW, e mostrou as características peculiares a este tipo de aplicação, os aspectos ligados à obtenção, tratamento e povoamento de DW, como a configuração do SGBD Oracle®, que precisou ser direcionada para grandes volumes de dados. O SCOPE mostrou que o entendimento das aplicações legadas, a maioria lotada em mainframes e escrita em linguagens antigas e o apoio total de profissionais ligados ao negócio e ao tecnológico era fundamental ao trabalho da equipe de desenvolvimento. Foram dois anos de trabalho que culminou em um DW que serve à organização e fornece informações estratégicas, atualizadas mensalmente, sobre as

saídas de recursos da Chesf. O SCOPE serviu de laboratório para a validação de alguns conceitos do Hydrus que foram aplicados na sua construção e mostrou que nem sempre grandes ferramentas são necessárias para a criação de bons DW.

O segundo projeto que se beneficiou dos fundamentos do ambiente Hydrus foi o GEAF, Gestão de Arrecadação e Faturamento, sistema concebido para processar o faturamento de vendas de energia da Chesf e controlar a arrecadação dos recursos referentes a estas vendas. O GEAF foi desenvolvido em 2001, com o apoio da equipe de negócio do NSOI-DEF e DFTE, dois departamentos da empresa cujas atividades estavam diretamente ligadas ao projeto. A grande novidade neste sistema era que ele não seguia os padrões normais para aplicativos pertencentes a sua categoria. Como a ANEEL, Agência Nacional de Energia Elétrica, vinha determinando mudanças nos contratos de venda de energia, um sistema para atender a elas precisava ser extremamente configurável. Por isto o GEAF precisava ser concebido como um ambiente capaz de ser estendido ao longo de sua vida, através de scripts e criação dinâmica de tabelas. Estas eram características oferecidas pelo ambiente Hydrus que serviu perfeitamente aos propósitos do projeto da Chesf. O Hydrus mostrou ao projeto GEAF que era possível construir um ambiente que fornecesse um conjunto de serviços, no caso ligados ao faturamento e arrecadação, que podia ser alterado conforme a necessidade, muitas vezes ampliando o número de funcionalidades oferecidas, tudo isto apenas trabalhando com scripts e tabelas virtuais. O GEAF vai completar quatro anos de existência e os conceitos usados na sua construção continuam válidos até o presente momento, mostrando que o ambiente Hydrus é atual e viável.

Com a experiência adquirida na construção do DW SCOPE e do GEAF, mais os estudos e pesquisas realizados sobre Data Warehouse, especificamente sobre o processo ETL, foi possível consolidar o Hydrus, um ambiente onde é possível definir uma ferramenta ETL capaz de executar as diversas tarefas ligadas ao processo de povoamento de DW ou de transferência de dados entre fontes diferentes, com tratamento do conteúdo movimentado.

5.3 ASPECTOS GERAIS

Como foram vistas nos capítulos anteriores, as características das ferramentas ETL são inúmeras e muitas são bastante complexas. Existem atualmente vários

produtos ETL, criados por empresas conceituadas no mercado ou em ambientes experimentais. São sistemas nascidos já há muitos anos, com muito tempo de mercado, tendo sido usados em diversos projetos de construção de DW. Mas estes produtos são normalmente caros e disponibilizam muitas funcionalidades que não são necessárias à maioria das aplicações de pequeno e médio porte de DW, tornando-se inadequadas. Por isto o ambiente Hydrus torna-se uma solução alternativa. Ele fornece a estrutura básica para que os usuários possam implementar o processo ETL de vários DW e possam reaproveitar e compartilhar as soluções construídas. As principais características do projeto Hydrus são:

- *Simplicidade*: o Hydrus é simples. A complexidade que ele possa alcançar será devido às implementações realizadas no seu ambiente e não às estruturas e funcionalidades que ele oferece. Sua proposta é disponibilizar o mínimo necessário à construção de uma ferramenta ETL e que todas as funcionalidades implementadas no seu ambiente possam ser utilizadas por outras construções.
- *Custo zero*: o Hydrus não é um ambiente comercial, com política de licenciamento pago. Ele, até por ter nascido como projeto de uma tese, adota as regras definidas para o código livre.
- *Extensão*: para que possa prover um ambiente onde ferramentas ETLs sejam concebidas, o Hydrus fornece os meios necessários à sua extensão como plugins, scripts.
- *Metadados*: para armazenar as definições das ferramentas ETL, bem como as funcionalidades e especificações do processo ETL que elas implementam, o Hydrus trabalha com um repositório de metadados centralizado. A administração e geração das informações deste repositório são feitas através de propriedades e funcionalidades das classes do ambiente.
- *Staging area*: a área de preparação de dados ou *staging area*, é usada em várias arquiteturas de DW, entre elas a *Top-Down* (INMON, 1997). A arquitetura do ambiente Hydrus faz uso desta área para gerar as tabelas temporárias e permanentes que são usadas durante o processo ETL.
- *Flexibilidade:* esta característica reflete a capacidade do ambiente Hydrus de alterar as suas funcionalidades em função de uma ferramenta ETL específica

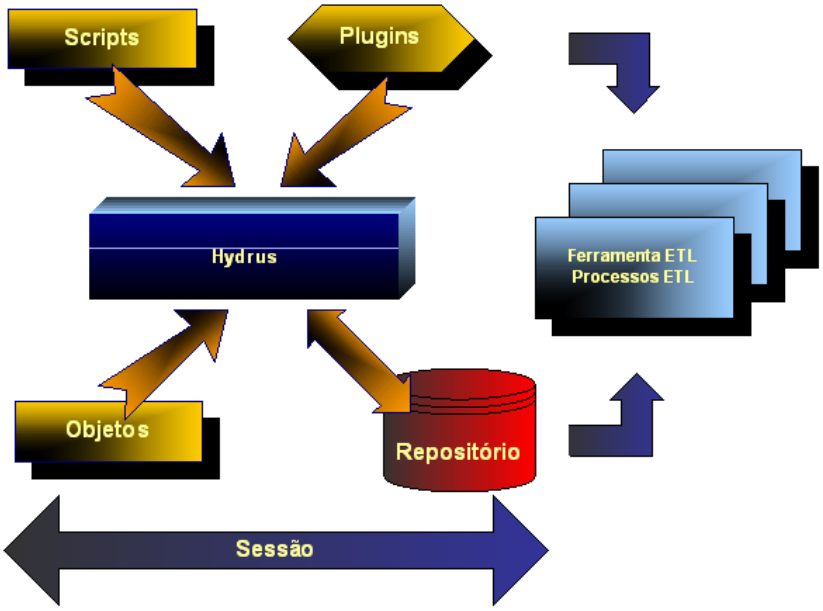
e seus processos, o que é operacionalizado através da identificação adequada dos requisitos que são implementados através de suas estruturas nativas como scripts e plugins.

- *Reusabilidade:* todas as definições, códigos e estruturas criadas no ambiente Hydrus, compreendendo uma ferramenta ETL e seus processos, são armazenadas em metadados reutilizáveis, compartilhados ou exportados para outros projetos.
- *Suporte a maioria das arquiteturas de DW*: como o ambiente Hydrus é aberto e extremamente configurável, ele não se prende a nenhuma arquitetura específica de DW. Ele fornece os meios necessários para que os programadores desenvolvam as funcionalidades adequadas à arquitetura em que eles estiverem trabalhando.

5.4 ARQUITETURA

O ambiente Hydrus foi concebido visando dar suporte à construção de qualquer tipo de ferramenta ETL que atenda às necessidades dos pequenos e médios projetos de DW. Ele é composto por um conjunto de classes através das quais se consegue implementar funcionalidades, que são armazenadas em um repositório central de metadados. Se as classes disponibilizadas no ambiente forem insuficientes, poder-se-á programar novas, que deverão ser descendentes da classe global clsObjeto. As novas classes serão incorporadas ao Hydrus como plugins. A figura 12, dá uma visão geral dos componentes do Hydrus.

Figura 12 - Arquitetura geral do ambiente Hydrus



Fonte: o autor (2004)

As classes fornecidas pelo Hydrus, mais as implementações e definições realizadas no seu ambiente, adicionadas aos plugins formam uma sessão que é armazenada em um repositório de metadados, em uma base de dados normalmente relacional. Cada sessão representa uma ferramenta ETL com suas funções e implementações como será visto adiante.

5.4.1 Conceitos gerais

O Hydrus é formado por um conjunto de pacotes que juntos são responsáveis por todas as funcionalidades do ambiente de apoio à construção de ferramentas ETL: objetos, acesso a dados, sessões, scripts, utilitários, segurança, configuração, relatórios, plugins.

- *Objetos:* este pacote contém as classes básicas do ambiente. Todos os objetos que são armazenados no repositório de metadados pertencem a classes descendentes deste pacote, bem como as incorporadas como plugins.
- *Acesso a dados:* este pacote contém as classes responsáveis pelas conexões nas diversas fontes de dados. Suas classes básicas fornecem uma forma genérica de se estabelecer ligações com os diversos provedores de dados, principalmente com os SGBDs;

- *Sessões:* este pacote contém a classe responsável pela gerência de uma sessão e a classe que controla o ambiente Hydrus.
- *Scripts:* uma forma de estender o ambiente é através de scripts que devem ser escritos em linguagem apropriada que dê ao programador acesso aos objetos definidos. Este pacote contém as classes que implementam os scripts do Hydrus.
- *Plugins:* este pacote contém as classes que gerenciam os plugins do ambiente. Elas definem como eles são carregados e disponibilizados, bem como são controlados e exportados, para que sejam compartilhados entre diversas sessões.
- *Utilitários:* este pacote armazena todas as classes utilitárias do sistema. Na realidade, vários pacotes podem ser classificados como utilitários.
- *Segurança:* este pacote contém as classes responsáveis pela implementação da segurança de uma sessão do Hydrus. Quando se deseja implementar um sistema alternativo de controle de acesso, isto deve ser feito a partir das classes deste pacote, via herança.
- *Configuração:* as configurações de cada sessão devem ser definidas através das classes e subclasses deste pacote.
- *Relatórios:* os relatórios criados no Hydrus são feitos através das classes definidas neste pacote e suas descendentes.

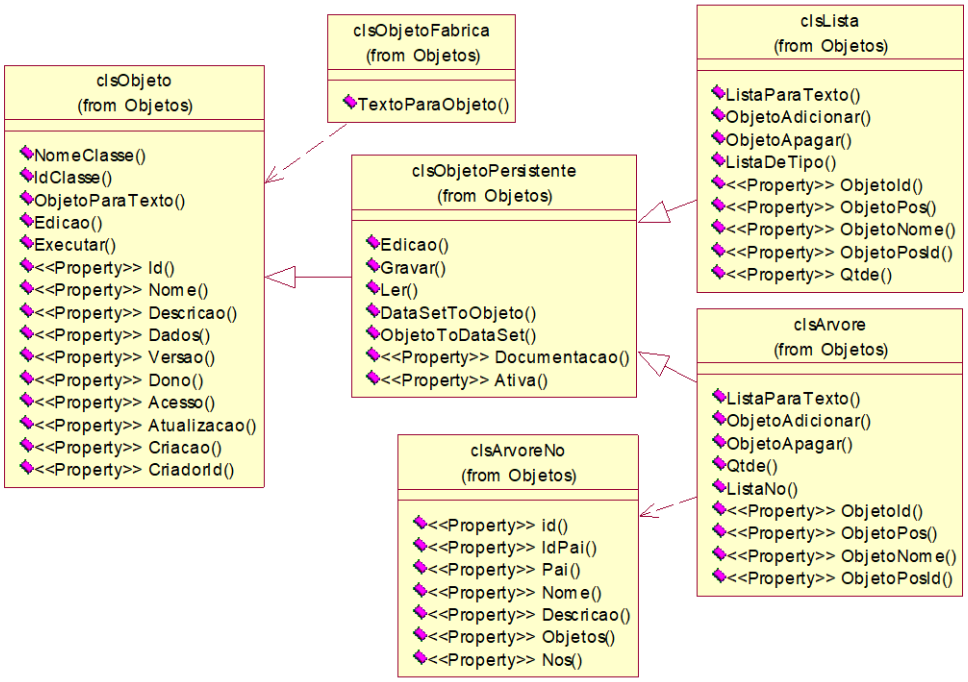
O número de pacotes deve crescer à medida que novas versões do Hydrus forem definidas, quando novas ideias serão adicionadas. A seguir cada um destes pacotes serão detalhados, sendo apresentados os conceitos gerais definidos para cada um deles. A dissertação não pretende definir minúcias sobre as classes dos pacotes, mas apresentar as ideias de cada uma delas. No final deste capítulo será descrito o protótipo experimental, criado a partir dos conceitos definidos para o ambiente Hydrus, com o objetivo de validar as ideias aqui apresentadas.

5.4.2 Pacote Objetos

O pacote Objetos contém as classes mais elementares do ambiente Hydrus das quais muitas outras descendem. Nele estão definidos vários aspectos importantes

como persistência e documentação. A figura 13 apresenta um diagrama de classes deste pacote.

Figura 13 - Classes do pacote Objetos do ambiente Hydrus



Fonte: o autor (2004)

5.4.2.1 Classe clsObjeto

A clsObjeto é a base de todas as classes fundamentais do ambiente Hydrus, como scripts, tabelas. Ela possui atributos que identificam o objeto e que são usados em diversas partes do ambiente. O quadro 2 apresenta a descrição destes atributos básicos:

Quadro 2 - Principais atributos da classe clsObjeto

Atributo	Significado
Id	Id é o identificador único do objeto;
Nome	Quando uma instância da classe clsObjeto é criada, é dado a ela um nome inicial baseado no Id. Todo objeto desta classe precisa ser identificado por um nome.
Descricao	Descrição do objeto;
Dados	Este atributo é coringa e pode ter diferentes significados de acordo com o tipo da classe. Ele armazena uma cadeia de caracteres;

VersaoDono	Estes atributos são utilizados pelo controle de versão do ambiente, quando implementado.
Acesso	Este atributo pode ser utilizado pelo sistema de segurança para implementar uma política de controle de acesso pontual, em relação a cada objeto instanciado no ambiente.
Atualizacao	Este atributo mantém a data e hora da última atualização sofrida pelo objeto
Criacao	Este atributo define a data e hora quando o objeto foi criado.
CriadorId	Este atributo contém o Id do usuário que criou o objeto.

Fonte: o autor (2004)

Cada instância da classe clsObjeto possui um valor único que o identifica, armazenado no atributo Id. É prevista em orientação a objetos a existência deste atributo, também chamado de OID ou *Object ID*. O ambiente Hydrus é o responsável pela geração dos valores do atributo ID e precisa garantir que eles são exclusivos. Como a implementação do ambiente vai depender de uma série de fatores como a linguagem, por exemplo, cabe ao analista definir como a geração dos valores do atributo Id será feita. Por exemplo, no protótipo foi usado um campo autoincremental de uma tabela exclusiva na geração dos valores.

A classe clsObjeto define alguns métodos importantes que são apresentados no quadro 3.

Quadro 3 - Principais atributos da classe clsObjeto

Método	**Significado**
NomeClasse	Este método de classe retorna o nome da classe
IdClasse	Este método de classe retorna um identificador único para a classe
ObjetoParaTexto	Este método é importantíssimo e retorna uma representação da classe em formato texto. Normalmente utiliza-se XML para representar a classe
Edicao	Este método, usado pela interface do ambiente, dispara a edição visual da classe.
Executar	Este método polimórfico responde de forma diferente de acordo com a classe que o implementa. Por exemplo, clsScript responde executando o script.

Fonte: o autor (2004)

De todos os métodos de clsObjeto, o mais importante é o *ObjetoParaTexto*. Ele retorna a representação texto do objeto. A classe clsObjetoFabrica utiliza esta representação para criar um clone do objeto ou reconstruí-lo. A figura 14 mostra a sequência de transformações.

Figura 14 - Transformações de clsObjeto – Texto - clsObjeto

Fonte: o autor (2004)

A representação texto dos objetos pertencentes à classe clsObjeto deve ser a mais apropriada à linguagem de desenvolvimento utilizada para implementar o ambiente Hydrus. No quadro 4 estão duas delas, uma em XML, adequada a Java, e outra, ao em Object Pascal, adequada ao Delphi.

Quadro 4 - Representação de um objeto em XML e em Delphi

Java

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1" class="hydrus.objetos.clsBotao">
    <object class="hydrus.objetos.clsBotao">
    <void property="caption">
        <string>Botão</string>
    </void>
    <void property="Width">
        <integer>129</string>
    </void>
    </object>
</java>
```

Delphi

```
object Botao: clsButton
  Caption = 'Botão'
  Width = 129
End
```

Fonte: o autor (2004)

5.4.2.2 Classe clsObjetoPersistente

Descendente direta de clsObjeto, esta classe implementa a persistência de objetos do ambiente Hydrus no repositório central de metadados. Toda classe que pode ser persistida é descendente direta de clsObjetoPersistente.

Dependendo do modelo de dados adotado para o repositório, várias técnicas podem ser empregadas para realizar a persistência. Neste trabalho sugere-se utilizar um banco de dados relacional para guardar os metadados, já que existem inúmeros SGBDs relacionais, tanto comerciais como gratuitos.

A persistência pode ser implementada através de várias técnicas. SCOTT (2004) discorre sobre o assunto e mostra como realizar o mapeamento de objetos para bancos de dados relacionais. Neste trabalho são apresentadas duas formas extremamente simples de realizar o mapeamento, uma das quais foi adotada pelo protótipo. Estas formas não contemplam o problema de dependências entre classes como a hierarquia, mas para o objetivo deste trabalho são adequadas, pois estas relações são estabelecidas pela aplicação à medida que os objetos são carregados.

5.4.2.2.1 Mapeamento direto para uma única tabela de Objetos

Uma forma simples de realizar a persistência dos objetos do ambiente Hydrus, num repositório relacional, é através de uma grande tabela com a estrutura base apresentada na figura 15:

Figura 15 – Definição da tabela Objetos

```
CREATE TABLE OBJETOS (
    OBJ_ID NUMBER(38, 0) NOT NULL,
    OBJ_NOME VARCHAR2(64) NOT NULL,
    OBJ_DESCRICAO VARCHAR2(196),
    OBJ_CRIADOR NUMBER(38, 0) NOT NULL,
    OBJ_DATA_CRIACAO DATE NOT NULL,
    OBJ_IMAGEM NUMBER(38, 0),
    OBJ_VERSAO NUMBER(38, 0),
    OBJ_ID_CLASSE NUMBER(38, 0),
    OBJ_DATA_ATUALIZACAO DATE,
    OBJ_ACESSO VARCHAR2(32),
    OBJ_DONO NUMBER(38, 0),
    OBJ_OBJETO LONG RAW
);
```

Fonte: o autor (2004)

Nesta forma, cada registro da tabela “Objetos” armazena uma instância da classe clsObjeto ou descendente dela. O campo OBJ_OBJETO, do tipo texto longo, é usado para guardar os dados da instância que é salva em sua representação texto.

Atributos comuns a todas as classes persistentes, como id, nome e descrição são salvos também como campos da tabela “Objeto”, pois isto facilita a busca dos objetos na base de dados.

Para salvar o objeto na base seguem-se os seguintes passos:

1. Obtém-se a representação texto do objeto através da execução do seu método ObjetoParaTexto;
2. Cria-se um registro na tabela “Objetos” e preenchem-se os seus campos com os atributos do objeto;
3. Atribui-se ao campo “OBJ_OBJETO”, do novo registro, o valor obtido da função ObjetoParaTexto;

Para se carregar o objeto da base faz-se o caminho inverso:

1. Pesquisa-se o objeto na tabela através de seu Id;
2. Obtém-se a representação texto do objeto a partir do campo “OBJ_OBJETO”;
3. Usa-se a classe clsObjetoFabrica para instanciar o objeto a partir da sua reapresentação texto obtida.

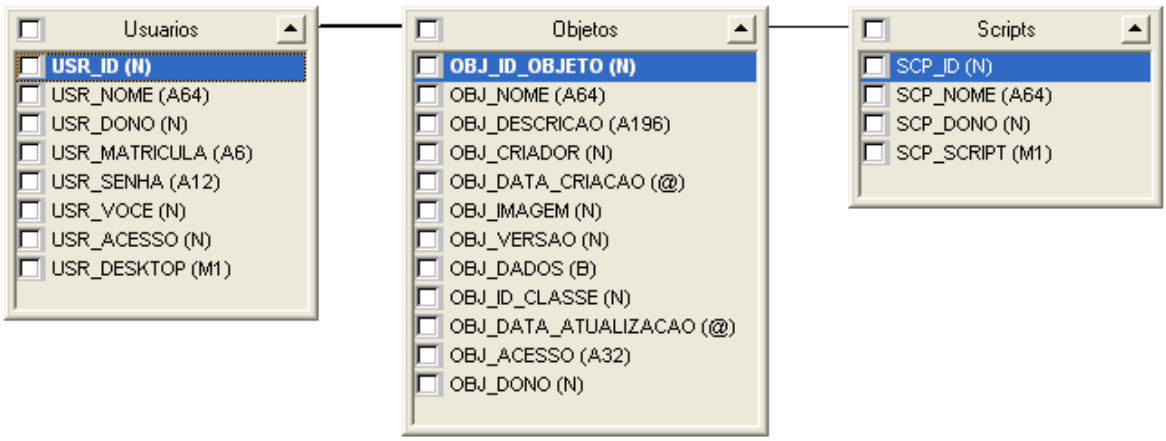
5.4.2.2.2 Mapeamento do objeto para campos de tabelas

Nesta forma de persistência, os atributos dos objetos são salvos como campos das tabelas relacionais. Esta forma foi adotada na construção do protótipo e, embora a programação dela foi mais trabalhosa, ela permitiu a utilização de todo o poder do SQL na manipulação dos dados armazenados dos objetos.

Para guardar os objetos do protótipo foi criada uma tabela para cada classe persistente e uma tabela geral nomeada de “Objetos”. Então se criou um relacionamento entre a tabela “Objetos” e cada uma das tabelas específicas das classes. Na figura 16 estão apresentadas, como exemplo, as tabelas referentes às classes que representam os Scripts e os Usuários.

A tabela “Objetos” armazena os atributos comuns a todas as classes persistentes, enquanto as outras tabelas guardam os valores dos atributos específicos. Embora não fosse obrigatória, a tabela “Objetos” foi criada para simplificar o manuseio dos objetos persistidos, pois ela centraliza os dados comuns a todos eles, como o Id, em um único lugar, o que facilita o trabalho das classes responsáveis por gerir o ambiente Hydrus.

Figura 16 - Transformações de clsObjeto – Texto - clsObjeto



Fonte: o autor (2004)

A classe clsObjetoPersistente contém quatro métodos e dois atributos adicionais em relação a sua antecessora clsObjeto. Todos os objetos pertencentes a classes descendentes dela podem ser armazenados no repositório de metadados. No quadro 5 são apresentados os seus métodos e atributos.

Quadro 5 - Principais métodos (M) e atributos (A) da classe clsObjeto

Método	Tipo	Significado
Gravar()	M	Este método grava o objeto no repositório;
Ler()	M	Este método lê os valores do objeto a partir do repositório;
DataSetToObjeto()	M	Durante o processo de leitura, este método é acionado para mapear e atribuir as informações vindas do repositório para o objeto;
ObjetoToDataset()	M	Durante o processo de gravação, este método é acionado para mapear e atribuir às tabelas do repositório os valores dos atributos do objeto;
Ativa	A	Por questão de performance, alguns objetos são lidos parcialmente do repositório e permanecem inativos. Quando eles são efetivamente utilizados, o restante da leitura é feito e o objeto é ativado.
Documentação	A	Este atributo guarda informações sobre o objeto e pode ser usado para implementar ajuda on-line e documentação. Cada implementação do Hydrus pode definir o nível de complexidade que desejar para este atributo.

Fonte: o autor (2004)

5.4.2.3 Classe clsLista e clsArvore

Tanto a classe clsLista como clsArvore implementam uma lista de objetos descendentes de clsObjeto. A primeira trabalha com listas simples, enquanto a segunda, com listas hierárquicas. Ambas são descendentes de clsObjetoPersistente e, portanto, suas instâncias podem ser persistidas no repositório de dados.

As listas definidas por estas classes possuem a capacidade de incluir, excluir, alterar ou pesquisar os objetos que guardam. Além disto, elas são otimizadas. Quando estão armazenadas no repositório de metadados e são carregadas, não trazem juntos os objetos que guardam, mas apenas os seus identificadores, Id. Só quando o objeto é requerido é que elas efetivamente carregam o objeto para a memória. Isto aumenta a performance porque diminui o overhead causado pela carga desnecessária de objetos.

As classes clsLista e clsArvore são úteis no ambiente Hydrus porque facilitam várias operações e padronizam a forma como os objetos são agrupados. No entanto, outros tipos de listas, desde que disponíveis, podem ser utilizados no ambiente Hydrus, tais como TStringlist, TList ou Vector.

5.4.3 Pacote Acesso a dados

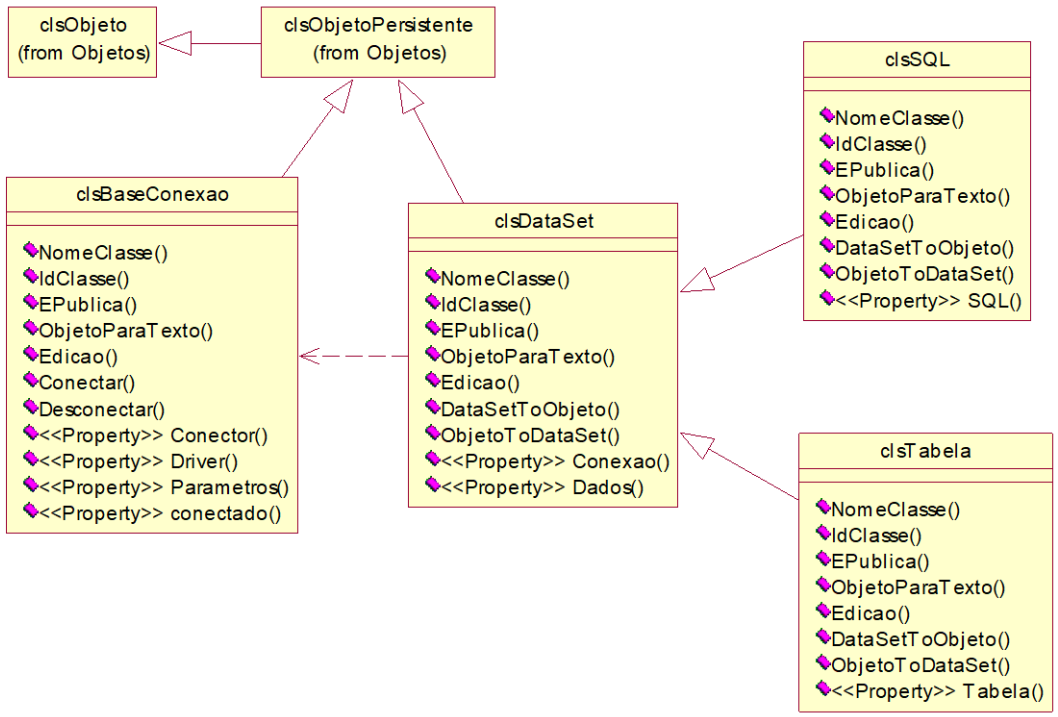
O pacote de acesso a dados possui as classes que possibilitam ao ambiente Hydrus manipular os dados das diversas fontes existentes, inclusive os metadados e a área de preparação. Seu principal objetivo é fornecer uma interface padronizada de acesso aos dados para as outras classes do Hydrus.

Basicamente este pacote possui duas categorias de classes: a responsável pela conexão com as bases e outra que obtém e manipula os dados. A primeira categoria fornece uma interface única para o estabelecimento de conexão com os diversos SGBDs e fontes de dados utilizados pela ferramenta ETL. A segunda permite que os dados das fontes sejam obtidos e modificados, fornecendo a capacidade de execução de sentenças SQLs, comandos nas linguagens do SGBD, acionamentos de *stored procedures* dos bancos relacionais, manipulação do conteúdo das tabelas.

Todas as classes deste pacote usam os componentes nativos, existentes nas linguagens usadas no desenvolvimento do Hydrus e, portanto, disponíveis nos scripts,

para fazer acesso aos dados. Isto significa que não foi reinventada uma biblioteca específica para manipular as fontes de dados, mas usadas as existentes. Por exemplo, no protótipo, usa-se o BDE, Borland® Database Engine, para trabalhar com as bases de dados. As classes do pacote de acesso a dados são apenas uma interface entre o Hydrus e estas bibliotecas nativas, funcionando como uma camada que padroniza o acesso às diversas bases de dados. A figura 17 mostra as classes deste pacote.

Figura 17 - Pacote de acesso aos dados



Fonte: o autor (2004)

5.4.3.1 Classe clsBaseConexao

A classe clsBaseConexao é responsável pelas conexões do ambiente Hydrus às bases de dados, sendo descendente direta de clsObjetoPersistente. Ela implementa a forma como é realizada a ligação com as fontes de dados. Para cada conexão com uma base de dados, deve existir uma instância desta classe. Durante a execução de um processo ETL é normal existirem várias instâncias de clsBaseConexao ou descendentes dela, uma para cada fonte de dados envolvida na operação. Incluem nesta lista também o repositório de metadados e a área de preparação.

Como esta classe descende de clsObjetoPersistente, todas as definições de conexões com as bases de dados envolvidas no processo ETL podem ser armazenadas no repositório central de metadados.

A classe clsBaseConexao funciona como uma fachada que fornece uma interface padronizada para conexão com as bases de dados. Seus principais métodos e atributos estão apresentados no quadro 6.

Quadro 6 - Principais métodos (M) e atributos (A) da classe clsBaseConexao

Método	Tipo	Significado
Conectar() Desconectar()	M	Estes métodos ativam ou desativam a conexão com a base de dados. Se a conexão falhar, o método gera uma exceção e o sistema poderá responder ao erro com o tratamento adequado.
Conector	A	Conector aponta para o objeto que efetivamente realiza a conexão com as bases de dados. Como clsBaseConexao é apenas uma interface para as classes do Hydrus, ela precisa de um objeto nativo da linguagem de desenvolvimento do ambiente para fazer a conexão.
Driver	A	Neste atributo é armazenado o tipo de banco de dados. Por exemplo, Oracle®, Posgres®.
Parametros	A	O atributo “Parametros”, no formato texto ou XML, armazena os a parâmetros que definem a conexão. Os valores deste atributo variam de acordo com o tipo de banco de dados a ser conectado.
Conectado	A	Este atributo informa o status da conexão com a base.

Fonte: o autor (2004)

A tecnologia empregada na conexão com as bases de dados vai depender da linguagem usada na implementação do Hydrus. Por exemplo, se for o Delphi, existem duas opções naturais: o BDE, ou Borland® Database Engine, que é considerada ultrapassada e o ADO. Para Java, existe um pacote fornecido também pela Borland®, o *DataExpress,* que pode ser uma boa escolha. Independente da tecnologia adotada é importante lembrar que clsBaseConexao precisa fazer uso dela para efetivamente estabelecer conexão com as bases de dados.

O atributo Conector aponta para o objeto pertencente à tecnologia escolhida que efetivamente realiza a conexão. Este objeto também fornece informações gerais sobre o banco de dados tais como *views*, tabelas, *procedures,* sequências.

5.4.3.2 Classe clsDataSet

clsDataSet é a base de todas as classes que acessam os dados. Diretamente dela descendem classes que manipulam as tabelas dos BD, que executam as *Stored Procedures* e as sentenças SQL dos bancos de dados relacionais. A conexão com as bases de dados é feita através de clsBaseConexao.

A classe clsDataSet é descendente de clsObjetoPersistente e por isto as suas definições podem ser persistidas no metadados. Como esta classe trabalha com tabelas, views, funções e índices, todas as informações sobre estas entidades podem ser armazenadas no repositório de metadados e posteriormente obtidas. Estas informações além de servirem aos processos ETLs, podem ser usadas na criação de documentação a respeito dos dados que são tratados no processo.

Existem basicamente dois atributos relevantes de clsDataSet: conexão e dados. O primeiro se refere ao objeto do tipo clsBaseConexao responsável por prover a conexão com as fontes de dados e que já foi visto na seção anterior. O segundo referencia o objeto utilizado no acesso aos dados. Este objeto é nativo da linguagem em que for implementado o ambiente Hydrus. Se porventura for o Delphi, ele pode ser o TTable, TQuery ou TAdoTable. No caso de Java, pode ser o TableDataSet ou QueryDataSet do pacote DataExpress.

Idêntico a classe clsBaseConexao, clsDataSet não implementa as funcionalidades que efetivamente fazem o acesso aos dados. Ao invés disto, ela utiliza os serviços providos por componentes nativos da linguagem usada no desenvolvimento do ambiente Hydrus. No caso do protótipo foram usados os serviços dos componentes do DataAcess, pacote fornecido pela Borland® que permite a conexão às bases de dados através do BDE – Borland Database Engine.

A classe clsDataSet é pai de todas as classes do ambiente Hydrus, que provêm acesso aos dados das bases usadas pela ferramenta ETL e seus processos.

5.4.3.3 Classe clsSQL

A classe clsSQL permite que se tenha acesso a uma ou mais tabelas no banco de dados usando sentenças SQL. Ela pode ser usada com todos os tipos de BD relacionais suportados pelo ambiente Hydrus tais como Sybase, SQL Server, entre

outros. O limite em relação aos bancos suportados vai depender da implementação realizada para o ambiente.

clsSQL é descendente direta de clsDataSet e herda todas as suas características, inclusive a capacidade de persistir no repositório de metadados. Esta classe pode executar qualquer sentença SQL, no entanto, ela não trata as diferenças existentes entre as sintaxes dos SQLs dos SGBDs. Uma alternativa a esta limitação seria o uso dos objetos do tipo clsTable, que será visto na próxima seção, ou o melhoramento desta classe permitindo que ela ajuste as sentenças SQL, no momento que forem executadas, de acordo com o tipo de banco de dados com o qual esteja trabalhando.

A classe clsSQL usa os componentes nativos das linguagens de desenvolvimento para executar as sentenças SQLs. No caso do Delphi, ela pode usar o *TQuery* ou TAdoQuery, enquanto em Java, QueryDataSet. Embora este texto apresente estes componentes como sugestão, outros podem ser empregados, dependendo apenas da escolha feita pela equipe que implementa o ambiente Hydrus.

O atributo mais importante de clsQuery é “SQL” que armazena a expressão SQL executada no processamento.

5.4.3.4 Classe clsTable

A classe clsTable faz o acesso a uma tabela de um banco de dados. Ela fornece acesso direto a cada registro e campos de uma tabela, não importando o tipo de BD. A independência fornecida por esta classe em relação às diferenças existentes entre os SGBDs pode ser um grande diferencial. Além disto, ela é uma classe informativa, pois fornece dados completos sobre as tabelas como quantidade de campos e seus tipos, índices, e outras características.

A classe clsTable, como clsQuery, usa os componentes nativos das linguagens de desenvolvimento para ter acesso a tabela de dados. No caso do Delphi, ela pode usar o *TTable* ou TAdoTable, enquanto em Java, TableDataSet, entre outras.

clsTable armazena informações sobre a tabela que ela acessa no atributo “Tabela”, do tipo texto. Este atributo pode simplesmente armazenar o nome da tabela ou informações completas sobre ela, no formato XML, por exemplo, dependendo da forma como o ambiente Hydrus é programado e qual o componente nativo é usado para a realização do acesso a tabela.

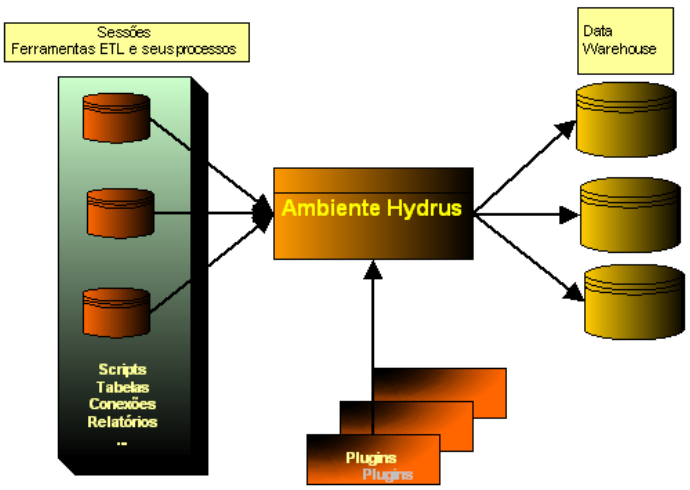
5.4.4 Pacote Sessões

Uma sessão representa uma ferramenta ETL, mais um processo ETL. Ela é armazenada no repositório de metadados, em uma base de dados. Cada sessão é guardada em um único banco de dados e não pode haver duas sessões no mesmo banco. Nas sessões estão contidas:

- Conexões
- Tabelas e SQLs
- Scripts
- Plugins
- Relatórios
- Segurança e controle de acesso
- Configurações
- Outros

Os objetos persistidos, pertencentes a classes descendentes de clsObjetoPersistente, são os elementos que juntos formam uma sessão ou, em outra visão, uma ferramenta ETL e seus processos. Quando se resolve compartilhar uma ferramenta, total ou parcialmente, está se compartilhando os seus objetos persistidos. Além disto, novas classes persistentes podem ser incorporadas ao ambiente Hydrus através de Plugins, o que permite a extensão do ambiente dentro de uma lógica e organização. A figura 18 mostra uma visão da arquitetura do ambiente Hydrus e seus componentes.

Figura 18 - Arquitetura do Hydrus



Fonte: o autor (2004)

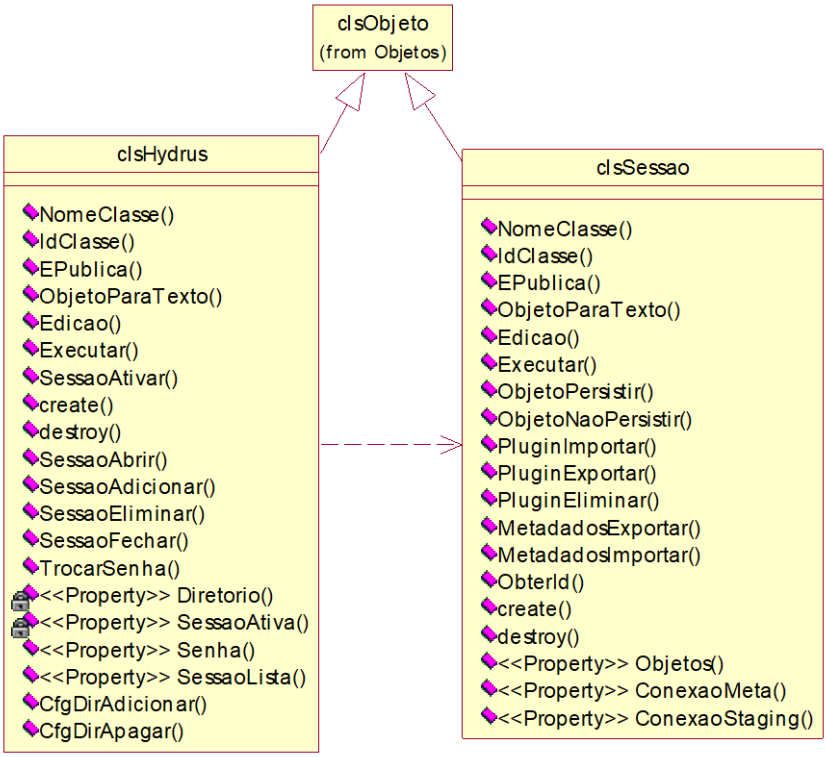
Esta arquitetura transforma o Hydrus em um ambiente capaz de apoiar a construção de ferramentas ETL, de forma ordenada e compartilhada. Como cada programador decide que plugins irá utilizar e quais scripts e objetos farão parte de sua sessão, ele pode criar diversas ferramentas, cada uma direcionada para um aspecto do processo ETL ou para um tipo específico de problema ligado ao povoamento de DW. Além disto, ele pode incorporar a sua ferramenta, componentes desenvolvidos por outros grupos de programadores simplesmente copiando dados entre repositório, o que facilita o seu trabalho sem a necessidade da instalação de pacotes de componentes.

Existem muitos ambientes de desenvolvimento capazes de incorporar plugins e componentes que são usados em suas linguagens de programação. No entanto, a proposta do Hydrus vai além disto, porque ela inclui a persistência dos elementos criados ou incorporados ao seu ambiente em bases de dados, que podem ser relacionais, orientadas a objeto, ou XML, armazenando em BD todos os artefatos tais como tabelas, scripts, que compõe a ferramenta ETL e seus processos. Por isto, todo um trabalho pode ser compartilhado simplesmente adicionando o conteúdo de uma base de dados à outra e uma ferramenta passa a ser definida através dos dados de uma base e não pelos códigos executáveis gerados a partir da compilação de um programa.

A ideia de armazenar em uma base de dados as definições das ferramentas ETL com seus formulários, wizards, códigos, eventos, e outros artefatos, facilita o gerenciamento dos projetos, o controle de versão e o trabalho compartilhado dos usuários.

Para gerenciar todas estas estruturas que compõe as ferramentas ETL e seus processos dentro do ambiente Hydrus foram definidas duas classes, clsHydrus e clsSessao cujas estruturas estão apresentadas na figura 19.

Figura 19 - Classes do pacote Sessão



Fonte: o autor (2004)

5.4.4.1 Classe clsHydrus

O ambiente Hydrus é controlado pela classe clsHydrus. Ela é responsável, entre outras coisas, pelo gerenciamento completo das sessões, possuindo métodos que permitem criá-las, eliminá-las, copiá-las. Além disto, clsHydrus controla todo o funcionamento do ambiente disparando os processos iniciais e realizando as configurações básicas necessárias.

Quando se executa o ambiente Hydrus, é criada uma instância da classe clsHydrus que faz três operações:

1. Inicia os objetos básicos de apoio e controle
2. Procura e carrega o arquivo de configuração
3. Baseado nas configurações abre uma sessão ou aguarda a seleção de uma pelo usuário.

O arquivo de configuração guarda as informações mais gerais sobre o ambiente. Ele, por exemplo, possui a lista de todas as sessões com as quais o ambiente trabalha. Além disto ele registra qual das sessões deve ser carregada automaticamente quando

o ambiente for executado e nela, quais as tarefas ou scripts devem ser disparados. Se não existir nenhuma sessão a ser carregada inicialmente, o ambiente espera pela seleção de uma pelo usuário.

O arquivo de configuração é escrito em XML, sendo gravado no mesmo local onde está o sistema Hydrus. Algumas vezes, uma parte do arquivo de configuração pode estar distribuída em diretórios compartilhados, na rede ou em URLs. Neste caso, após carregar o arquivo de configurações, o objeto clsHydrus obtém os endereços compartilhados e busca as informações complementares.

É prevista a definição de uma senha de acesso para os arquivos de configurações, que deverá ser pedida no momento da sua gravação e leitura. Opcional, a execução deste requisito implicaria na criptografia dos arquivos de configurações.

Após ler o arquivo de configurações e seus complementos, o ambiente Hydrus tem a relação de todas as sessões disponíveis. O ambiente está pronto para entrar em operação. A classe clsHydrus implementa uma série de métodos que gerencia o arquivo de configuração e seus complementos.

As sessões são registradas nos arquivos de configurações em listas de elementos XML. Como elas possuem informações sigilosas como as senhas de acesso aos BDs onde estão armazenadas, elas fazem a criptografia de suas informações antes de as disponibilizarem para os arquivos de configurações.

Os principais métodos e atributos da classe clsHydrus estão descritos no quadro 7, que foi simplificada para evitar detalhamentos desnecessários.

Quadro 7 - Principais métodos (M) e atributos (A) da classe clsHydrus

Método	Tipo	Significado
SessaoAbrir(), SessaoAdicionar(), SessaoEliminar(), SessaoFechar()	M	Adiciona, elimina, abre e fecha uma sessão do Hydrus. Ao ser adicionada uma sessão, deve ser informado o local, no arquivo de configurações, onde ela será salva.
SessaoAtivar()	M	Este método ativa uma determinada sessão
CfgDirAdicionar() CfgDirApagar()	M	Adiciona e elimina arquivos complementares de configuração. Outros métodos ligados aos arquivos de configurações podem ser definidos no ambiente.
Senha	A	Este atributo é utilizado na criptografia dos arquivos de configurações.

SessaoLista	A	Lista todas as sessões disponíveis no ambiente

Fonte: o autor (2004)

5.4.4.2 Classe clsSessao

Enquanto clsHydrus administra o ambiente Hydrus como um todo, clsSessao administra o projeto de uma ferramenta ETL. Esta classe é responsável pela guarda dos artefatos que representam a ferramenta e o processo ETL que ela implementa. A classe clsSessao controla uma série de objetos que são armazenadas em um repositório central de metadados. Este repositório, a princípio é gravado em um banco de dados preferencialmente relacional, mas dependendo como o ambiente Hydrus é implementado, pode-se usar outros tipos de estrutura de armazenamento como arquivos XML ou planilhas.

clsSessao também trabalha com uma área temporária de dados onde armazena informações transitórias ou permanentes, dependendo de como o processo ETL e os objetos que o implementam estão construídos. Esta área está explicitamente definida no ambiente porque ela é útil à classe clsSessao que a usa para suas próprias necessidades como armazenamento temporário de arquivos XML, criação de tabelas transitórias durante a edição de seus objetos persistentes. Esta área também está disponível para a ferramenta ETL e seus processos.

Cada ferramenta ETL implementada no ambiente Hydrus é contida inteiramente em uma sessão, bem como todos os seus componentes. A classe clsSessao controla e administra os seguintes componentes:

- Todos os objetos persistentes que compõe a ferramenta ETL e o processo que ela implementa;
- Todas as bibliotecas, plugins, usadas pela ferramenta;
- O repositório central de metadados;
- A área de armazenamento temporário e preparação.

5.4.4.2.1 Objetos persistentes

A classe clsSessao administra todos os objetos persistentes, ou seja, aqueles pertencentes a classes descendentes de clsObjetoPersistente e armazenados como metadados. Ela mantém uma lista contendo todos estes objetos e dá aos scripts

acesso total a eles. Com isto, é possível criar, apagar ou alterar qualquer objeto do ambiente Hydrus através de um script. Esta capacidade cria uma gama de possibilidades que pode ser explorada pelos programadores. Por exemplo, é possível criar *wizards* que geram novos scripts, ou cadastrar uma dezena de usuários a partir de dados lidos de uma planilha Excel, tudo feito através de scripts.

Quando um objeto pertencente às classes descendentes de clsObjetoPersistente é criado no ambiente, ele não é automaticamente armazenado no metadados. Existem dois métodos da classe clsSessao, descritos posteriormente, que fazem a persistência ou a eliminação destes objetos. Isto significa que, durante a execução de um script, no ambiente Hydrus, é possível criar instâncias temporárias de objetos descendentes de clsObjetoPersistente. Por exemplo, pode-se gerar um script temporário, executado uma única vez e posteriormente descartado.

O detalhamento de uma proposta de interface para o ambiente Hydrus está fora do escopo desta dissertação, mas recomenda-se que a lista de objetos persistidos seja apresentada em uma janela onde os referidos objetos fiquem dispostos em uma estrutura de árvore. Além disto, recomenda-se também que todas as configurações ligadas à interface sejam armazenadas no metadados, de forma que o usuário do Hydrus possa organizar os seus objetos como melhor lhe convier e seus ajustes sejam persistidos.

5.4.4.2.2 Biblioteca de Plugins

Todos os plugins usados em uma sessão são administrados pela classe clsSessao que possui métodos para importá-los, exportá-los e eliminá-los. Eles são mantidos no repositório de dados como bibliotecas e carregados para o ambiente Hydrus sempre que a sessão é ativada. É recomendada a implementação de controle de versão para os plugins.

5.4.4.2.3 Repositório central de metadados

O repositório de metadados guarda todos os objetos persistidos, além dos plugins e informações complementares como configurações da interface visual e preferências dos usuários. A classe clsSessao define um atributo, do tipo clsBaseConexao, que mantém uma conexão permanente com o repositório de

metadados enquanto a sessão estiver ativa. Pode-se dizer que o repositório de metadados do ambiente Hydrus guarda todas as informações sobre a ferramenta ETL e o processo que ela implementa tais como conexões com as bases, scripts que executam as atividades, formulários que compõe a interface, entre outras. A sessão está totalmente descrita nos seus dados.

A forma como os objetos são persistidos no repositório de metadados depende de como o ambiente Hydrus é implementado. Pode se usar uma única tabela de um banco relacional para guardar todos os objetos, por exemplo. Uma das vantagens da arquitetura do Hydrus é que ela possibilita o trabalho concorrente de programadores porque o repositório é centralizado. Da mesma forma, ela permite que diversos processos ETL sejam executados em paralelo. A arquitetura do Hydrus também possibilita a criação de dois ambientes distintos, de produção e de desenvolvimento, referente à mesma ferramenta e processo ETL, já que é possível duplicar os metadados.

5.4.4.2.4 A área de armazenamento temporário e preparação;

Finalmente, a área de armazenamento temporário ou preparação de dados é usada para diversos fins pela sessão. Ela é principalmente utilizada durante a execução do processo ETL, mas também pode ser usada em outras operações como na edição de objetos persistentes ou exportação de metadados.

Uma sessão ativa mantém permanentemente uma conexão com a base de dados da área temporária. Esta conexão só termina quando a sessão é fechada. Durante a abertura da sessão, se houver falha na conexão, o processo é interrompido e a sessão permanece fechada.

Os principais métodos e atributos da classe clsSessao estão listados no quadro 8:

Quadro 8 - Principais métodos (M) e atributos (A) da classe clsSessao

Método	Tipo	Significado
PluginImportar() PluginExportar() PluginEliminar()	M	Estes métodos realizam a importação, exportação e eliminação de um conjunto de plugins do ambiente Hydrus.
MetadadosExportar()	M	Estes métodos fazem a exportação e importação de informações dos metadados. Eles trabalham ao

MetadadosImportar()		nível de classes, objetos e definem um conjunto de parâmetros que dá total controle sobre estas operações.
ObjetoPersistir()	M	Este método torna persistente um conjunto de objetos pertencentes a classes descendentes de clsObjetoPersistente.
ObjetoNaoPersistir()	M	Este método retira do repositório de metadados um conjunto de objetos persistidos.
ObterId()	M	Este método gera valores nunca repetidos usados na criação dos OID dos objetos.
ConexaoMeta	A	Aponta para o objeto que mantém uma conexão com o repositório de metadados.
ConexaoStaging	A	Aponta para o objeto que mantém uma conexão com a área temporária de dados.
Objetos	A	Este atributo aponta para uma lista do tipo clsArvore que mantém todos os objetos persistidos da sessão. Como a classe clsArvore otimiza a manutenção de suas listas, os objetos que ela guarda só são carregados da base de dados sobre demanda.
Ativa	A	Este atributo permite a ativação ou desativação de uma sessão.

Fonte: o autor (2004)

5.4.5 Pacote Scripts

Os scripts tornam o ambiente Hydrus programável e extensível. Através deles, todas as funções responsáveis pelo processo ETL podem ser definidas e executadas. Eles são pequenos programas que realizam um conjunto de tarefas no ambiente e implementam as funcionalidades de uma ferramenta ETL ou de um processo ETL.

A linguagem de programação utilizada nos scripts deve ser a mesma adotada no desenvolvimento do ambiente Hydrus. A justificativa está no fato de que linguagens idênticas fornecem maior integração entre o ambiente e os scripts. Se isto não for possível, deve-se escolher a linguagem mais próxima ou compatível.

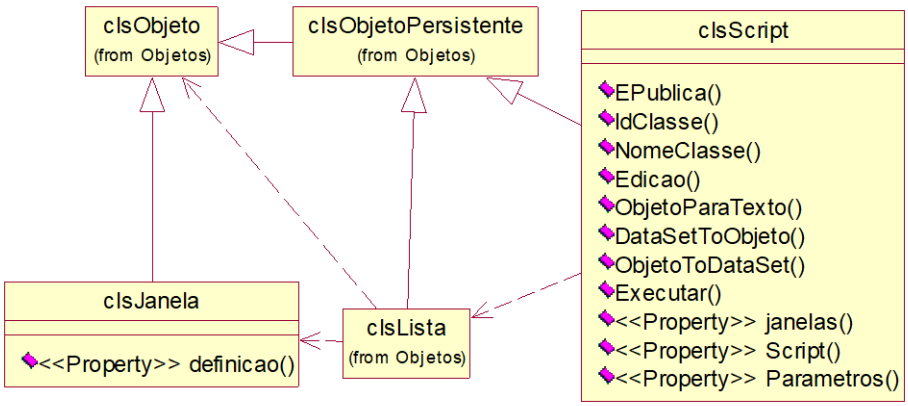
A implementação do ambiente Hydrus deve prover os scripts com algumas características importantes. Os scripts devem:

- Acessar todos os objetos persistentes do ambiente;
- Acessar todas classes e bibliotecas públicas contidas nos plugins incorporados às sessões;

- Ter acesso a uma rica biblioteca de classes nativas do ambiente;
- Possuir um editor apropriado com características que facilitem a codificação, como sintaxe colorida;
- Ter a capacidade de fazer chamada a outros scripts;
- Suportar os elementos básicos das linguagens de programação como loops, variáveis, funções;
- Ser interpretado ou compilado, com um analisador léxico e de sintaxe;
- Poder fazer chamada a aplicativos externos;
- Possuir um depurador.

Quanto mais agradável e cheio de recursos for o ambiente de desenvolvimento provido pelo Hydrus para a programação dos scripts, mais produtivo será o trabalho dos programadores. Por isto, deve ser dada muita atenção à implementação dos editores de scripts. Alguns editores específicos devem ser fornecidos pelo ambiente, como os usados na criação e manutenção de formulários ou os *wizards* para geração automática de códigos. A figura 20 apresenta as classes do pacote Script.

Figura 20 - Classes do pacote Script



Fonte: o autor (2004)

5.4.5.1 Classe clsScript

A classe clsScript, descendente de clsObjetoPersistente, é responsável por implementar as funcionalidades referentes aos scripts do ambiente Hydrus. Através dela o programador cria os scripts que executam os processos ETL e definem a ferramenta ETL.

A classe clsScript possui as seguintes propriedades definidas no quadro 9:

Quadro 9 - Principais métodos (M) e atributos (A) da classe clsScript

Método	Tipo	Significado
Executar()	M	Este método executa o script. Antes da execução, todos os parâmetros são passados para o script. Após a execução, os parâmetros de retorno são devolvidos, junto com informações sobre os erros, caso eles tenham acontecido.
Editar()	M	Este método aciona o editor gráfico do script.
Script	A	Este atributo guarda o script e informações adicionais sobre ele
Parâmetros	A	Este atributo armazena os parâmetros de entrada e saída
Janelas	A	O ambiente Hydrus deve fornecer um editor de formulários. Este atributo guarda as definições de formulários do script.

Fonte: o autor (2004)

5.4.5.2 Classe clsJanela

Se os scripts trabalham com formulários, eles têm que fornecer um editor para eles. Trata-se de um programa com funcionalidades direcionadas à criação de janelas de entrada e saída de informações.

Além disto, os formulários editados devem estar acessíveis nos códigos dos scripts. Algumas linguagens de programação permitem que um script ou unidade de código seja associado a apenas um formulário. É o caso do Delphi onde cada unit, extensão “PAS”, pode ser ligada a um formulário, extensão “DFM”. No ambiente Hydrus é interessante que esta restrição não aconteça de forma tal que um script possa estar associado a vários formulários.

A classe clsJanela implementa um formulário no ambiente Hydrus. Ela fornece um editor gráfico que permite a definição do formulário que pode ser utilizado no script. Cada script pode ter acesso a vários formulários.

5.4.5.3 Classe clsTarefa

Uma classe útil ao processo ETL é o gerenciador de tarefas. A sua função é controlar a execução de um conjunto de scripts, disparando-os na ordem correta, controlando a sequência em que eles são executados. Além disto, esta classe gerencia as dependências entre scripts e permite a execução paralela deles.

clsTarefa é uma classe que pode ser implementada tanto como descendente de clsObjetoPersistente, como de clsObjeto. No primeiro caso, ela ganha luz própria e pode existir independente de qualquer script. Além disto, ela passa a ser editada visualmente. No segundo caso, ela é instanciada, configurada e executada dentro de um script e sua edição visual vai depender dos editores fornecidos pela classe clsScript.

5.4.6 Pacote Plugins

Os plugins são componentes incorporados ao ambiente do Hydrus que aumentam as suas funcionalidades. Qualquer ambiente que se proponha a ser extensível precisa suportar plugins. No ambiente Hydrus, eles são adicionados às sessões e passam a estar disponíveis para os programadores de scripts. A classe clsSessao possui métodos que fazem a carga ou exportação de plugins, além da sua exclusão.

Existem vários modelos para desenvolvimento e incorporação de um plugin ao ambiente. Como eles são códigos binários, o modelo a ser adotado vai depender da linguagem na qual o Hydrus é construído. Sendo o Delphi, por exemplo, pode-se adotar o modelo baseado em DLL, padrão definido pela Microsoft® que incorpora funcionalidades aos sistemas através de bibliotecas dinâmicas. Outro modelo é baseado em ActiveX e COM+, outro padrão criado pela Microsoft® que trata os plugins como componentes distribuídos. Em Java, as classes depois de compiladas podem ser dinamicamente carregadas, sendo, portanto, mais natural a implementação de plugins nesta linguagem. Por exemplo, um modelo consagrado de utilização de plugins em Java é o projetado para o ambiente Eclipse (IBM, 2003). Trata-se de uma plataforma criada para fornecer um ambiente integrado de desenvolvimento que pode ser usado para criar aplicações em diversas linguagens tais como Java, C++, entre outras. Segundo a própria documentação, Eclipse é uma plataforma que serve para qualquer coisa e nada especificamente. Esta afirmação se baseia, principalmente, na sua capacidade de incorporar plugins para qualquer finalidade.

Independente da forma como os plugins são implementados no ambiente Hydrus, eles e seus componentes precisam:

- Poder ser usados nos scripts
- Poder ser persistidos no repositório de metadados

- Ter acesso aos demais objetos públicos do ambiente
- Ser facilmente incorporados e controlados

O controle dos plugins no ambiente Hydrus é realizado pela classe clsSessao, vista anteriormente. Ela possui métodos que carregam, persistem, descarregam e controlam as versões dos plugins.

5.4.7 Pacote Utilitários

O pacote “utilitários” é padrão em todos os sistemas. Ele fornece classes cujas funcionalidades são gerais e apoiam as diversas tarefas do ambiente. Por exemplo, classes que tratam a criptografia, XML e acesso ao sistema de diretórios.

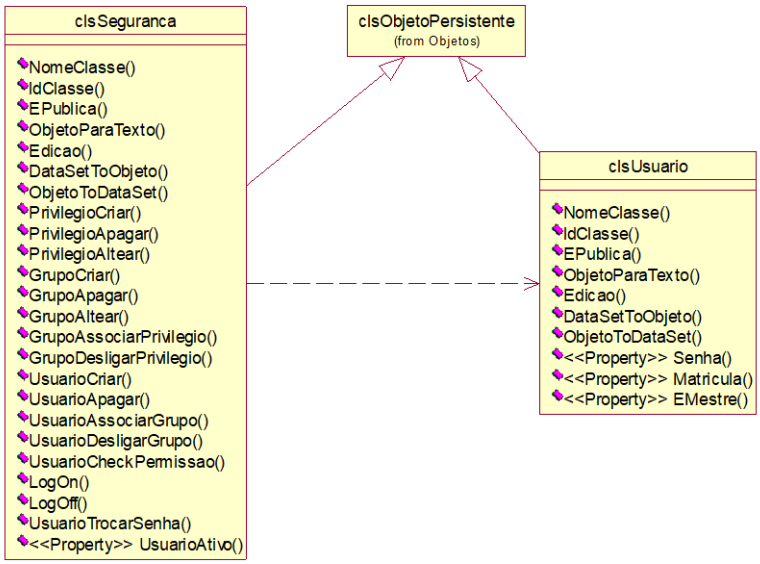
Classes deste pacote prestam serviços a todas as outras e existe uma infinidade delas.

5.4.8 Pacote Segurança

A segurança se refere a todas as funcionalidades ligadas ao controle de acesso do ambiente, incluindo privilégios, grupos de acesso e cadastros dos usuários. O Hydrus fornece um pacote padrão que trata deste assunto, no entanto implementações alternativas podem ser criadas. Basicamente duas classes estão ligadas à segurança conforme é apresentado na figura 21: clsUsuario e clsSegurança.

O sistema de segurança do Hydrus é associado às sessões ao invés do ambiente. Isto significa que cada sessão possui suas próprias definições de segurança como lista dos usuários com acesso, grupos e privilégios.

Figura 21 - Classes do pacote Segurança



Fonte: o autor (2004)

5.4.8.1 Classe clsSeguranca

clsSeguranca é responsável pelo controle de acesso no ambiente Hydrus. Descendente direto de clsObjetoPersistente, ela e suas classes de apoio são persistidas no repositório. A definição de clsSegurança pode ser tão complexa quanto o analista responsável pela implementação do Hydrus deseje. Um sistema básico de segurança deve, no entanto, possuir:

- Cadastro de grupos de acesso
- Cadastro de privilégios ou permissões
- Controle de senha e login e logoff
- Cadastro de usuários e associação deles a grupos de acesso

Embora não seja intenção deste texto detalhar todo um sistema de controle de acesso, mesmo porque existem vários disponíveis que poderiam ser analisados e adotados no ambiente, apresenta-se no quadro 10 uma proposta extremamente simples de métodos e atributos para a classe clsSeguranca. As classes de apoio como clsPrivilegios, clsGrupos foram omitidas.

Quadro 10 - Principais métodos (M) e atributos (A) da classe clsSeguranca

Método	Tipo	Significado
PrivilegioCriar PrivilegioApagar PrivilegioAlterar	M	Estes métodos gerenciam os privilégios das sessões realizando operações básicas sobre eles. Privilégios são as permissões definidas no sistema de segurança. Por exemplo, “criar script”.
GrupoCriar GrupoApagar GrupoAlterar	M	Estes métodos gerenciam os grupos de acesso das sessões realizando as operações básicas sobre eles. Grupos de acesso agrupam privilégios facilitando a atribuição deles aos usuários.
GrupoAssociarPrivilegio GrupoDesligarPrivilegio	M	Estes métodos fazem a associação e desassociação entre um grupo e um privilégio.
UsuarioCriar UsuarioApagar UsuarioEditar	M	Estes métodos gerenciam os usuários das sessões realizando as operações básicas sobre eles
UsuarioAssociarGrupo UsuarioDesligarGrupo	M	Estes métodos fazem a associação e desassociação entre um grupo e um usuário
UsuarioChecaPermissao	M	O método UsuarioChecaPermissao checa se um usuário tem permissão para realizar determinada operação.
UsuarioTrocarSenha	M	Este método realiza a alteração da senha do usuário ativo
LogOn LogOff	M	Estes métodos fazem o LogOn e LogOff das sessões
UsuarioAtivo	A	Este atributo referência o usuário ativo em um dado instante

Fonte: o autor (2004)

5.4.8.2 Classe clsUsuario

O princípio elementar de um controle de acesso é gerenciar as permissões dos usuários nos sistemas. Esta classe, persistente, implementa as funcionalidades referentes aos usuários das sessões do ambiente Hydrus.

Os principais atributos estão apresentados na quadro 11.

Quadro 11 - Principais atributos da classe clsUsuario

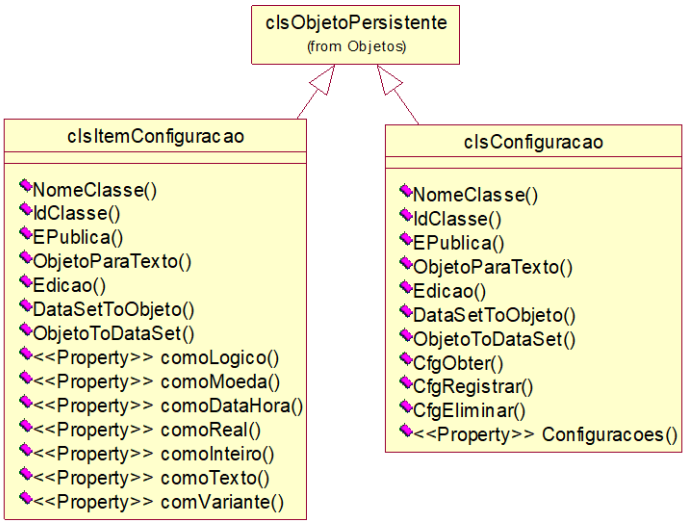
Atributo	Significado
Senha	Armazena a senha do usuário, criptografada. A criptografia da senha evita a sua divulgação indevidamente
Matricula	O Hydrus pode armazenar informações complementares dos usuários como matrícula, cpf, identidade. Este atributo pode ser usado para este fim.
EMestre	Este atributo define se o usuário é máster na sessão. Este é um artifício usado para rapidamente prover um usuário com todos os privilégios da sessão.

Fonte: o autor (2004)

5.4.9 Pacote Configuração

O pacote de configuração fornece uma solução simples e adequada para o registro de configurações das sessões do ambiente Hydrus. Basicamente existe uma classe que controla as configurações, clsConfiguracao que são armazenadas em objetos do tipo clsItemConfiguração, conforme é mostrado na figura 22.

Figura 22 - Classes do pacote Configuração



Fonte: o autor (2004)

Cada configuração é armazenada como uma instância da classe clsItemConfiguracao que trabalha com valores do tipo lógico, moeda, data e hora, real, inteiro, texto e variante. O último tipo é genérico e pode ser usado para armazenar imagens, binários.

A classe clsConfiguracao possui basicamente três métodos e uma propriedade relevante apresentados no quadro 12.

Quadro 12 - Principais métodos (M) e atributos (A) da classe clsConfiguracao

Método	Tipo	Significado
CfgObter()	M	Este método devolve um item do tipo clsItemConfiguracao através do qual pode-se obter o valor da configuração
CfgRegistrar()	M	Este método cria um item de configuração
CfgEliminar()	M	Este método elimina um item de configuração
Atributo		Significado
Configurações	A	Este atributo segura a lista de configurações de uma sessão

Fonte: o autor (2004)

5.4.10 Pacote Relatórios

Relatório é um artefato básico de todo os sistemas. Trata-se de um material informativo que apresenta dados formatados em planilhas, gráficos. O ambiente Hydrus tem uma classe para relatórios, a clsRelatorio, mas não possui um gerador de relatórios, pois está fora de escopo fornecê-lo. Ao invés, o ambiente Hydrus utiliza tecnologias consagradas na área como o Crystal Report que é fornecida pela Seagate, entre outras. No protótipo, foi usado este produto.

5.4.10.1 Classe clsRelatorio

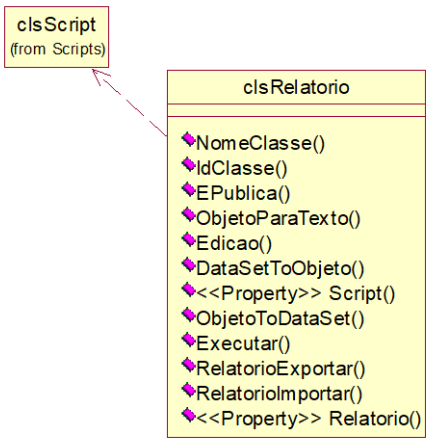
No ambiente Hydrus, a classe responsável pelos relatórios é clsRelatorio. Ela gera os relatórios em duas fases:

1. Execução do script de preparação dos dados que serão usados no relatório
2. Geração e apresentação do relatório através do acionamento do gerador de relatórios

A execução de um script antes da realização do relatório se faz necessária para que os dados sejam preparados dentro do ambiente do Hydrus. Desta forma, pode-se aproveitar as conexões, tabelas e outros scripts disponibilizados no ambiente e que não estariam facilmente acessíveis no gerador de relatório.

Como o ambiente Hydrus não define um gerador de relatório específico, a escolha de um é livre, desde que ele possa ser incorporado ao ambiente. Algumas sugestões são o Quickreport e o Seagate Crystal Report. O trabalho da classe clsRelatorio é feito usando o princípio da delegação. A classe guarda os dados do relatório, mas delega ao gerador a tarefa de executá-lo. A figura 23 apresenta os métodos e atributos da classe clsRelatorio.

Figura 23 - Classe clsRelatório



Fonte: o autor (2004)

Os principais métodos e atributos da classe clsRelatorio estão apresentados no quadro 13.

Quadro 13 - Principais métodos (M) e atributos (A) da classe clsRelatorio

Método	Tipo	Significado
Executar()	M	O método executar constrói e apresenta o relatório.
RelatórioExportar() RelatorioImportar()	M	Como a classe clsRelatorio armazena os dados do relatório que é executado efetivamente por um gerador, ela fornece estes dois métodos para exportar e importar estes dados.
Relatorio	A	Este atributo guarda os dados do relatório. Normalmente são informações binárias
Script	A	Este atributo aponta para o script que é acionado no momento da execução do relatório

Fonte: o autor (2004)

5.5 ESTUDO DE CASO – PROTÓTIPO EXPERIMENTAL

5.5.1 Introdução

Embora os aspectos apresentados nas seções anteriores tenham sido corroborados pela experiência adquirida ao longo dos anos, pelos projetos reais desenvolvidos em empresas de grande porte e por estudos realizados a respeito das características referentes a DW, a implementação de um protótipo é fundamental para o reforço e validação dos conceitos. Por esta razão, a partir do ambiente global do Hydrus foram selecionados e implementados os aspectos considerados mais relevantes e básicos para a validação da proposta da dissertação.

O protótipo foi construído no ambiente de desenvolvimento Delphi, da Borland®, em Object Pascal. Embora outras linguagens oferecessem boas funcionalidades e opções para o desenvolvimento do protótipo, como Java, a experiência e familiaridade adquirida ao longo de vários anos nesta ferramenta foram decisivas na escolha.

5.5.2 Considerações Gerais

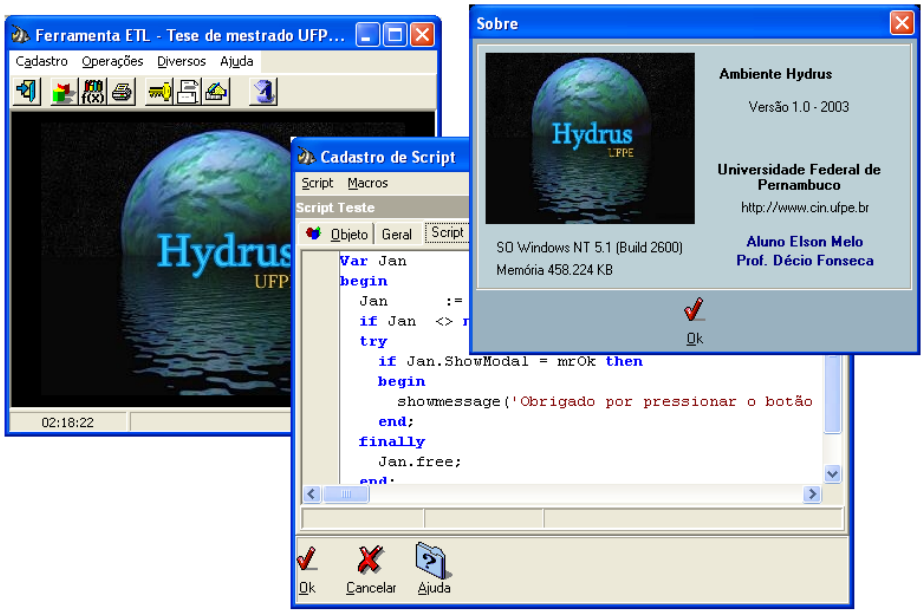
Na concepção do protótipo, o esforço necessário para implementação de cada aspecto apresentado neste trabalho foi avaliado. Algumas funcionalidades foram deixadas para uma versão posterior, enquanto outras foram simplificadas. O principal objetivo do trabalho foi provar a viabilidade da implementação de um ambiente extensível que serve de apoio à construção de ferramenta ETL através de trabalho ordenado.

Para apresentar o trabalho desenvolvido, esta seção seguirá a mesma sequência adotada na apresentação dos conceitos do ambiente Hydrus, fazendo um comparativo entre cada aspecto com a solução adotada no protótipo.

Como o ambiente Hydrus foi programado em Object Pascal, as classes implementadas estão seguindo o padrão sugerido pela Borland®, que diferem do adotado neste documento. Por exemplo, todas as classes em Object Pascal são normalmente começadas pela letra T, TObject, Ttable, enquanto nesta dissertação definiu-se que elas começariam por “cls”.

A interface do ambiente Hydrus é um aspecto que não foi abordado nas seções anteriores porque foi considerado um tema fora do escopo da dissertação. Evidentemente, quando se desenvolve um aplicativo, ela é definida naturalmente. Por isto, durante as próximas seções, características ou detalhes da interface serão apresentados para que o entendimento do protótipo seja facilitado, sempre que for preciso.

Figura 24 - Protótipo Experimental Hydrus



Fonte: o autor (2004)

5.5.3 Objetos básicos

A implementação do protótipo seguiu as especificações definidas para o ambiente Hydrus. Algumas simplificações, no entanto, foram feitas porque não era objetivo do projeto construir o ambiente completo. As classes clsObjeto e clsLista foram implementadas integralmente, mas mudanças importantes foram feitas como a classe clsObjetoPersistente que não foi programada por causa da simplicidade do protótipo. Ao invés disto, suas características foram transferidas para a classe superior, clsObjeto, que na implementação foi batizada de TelsObjeto.

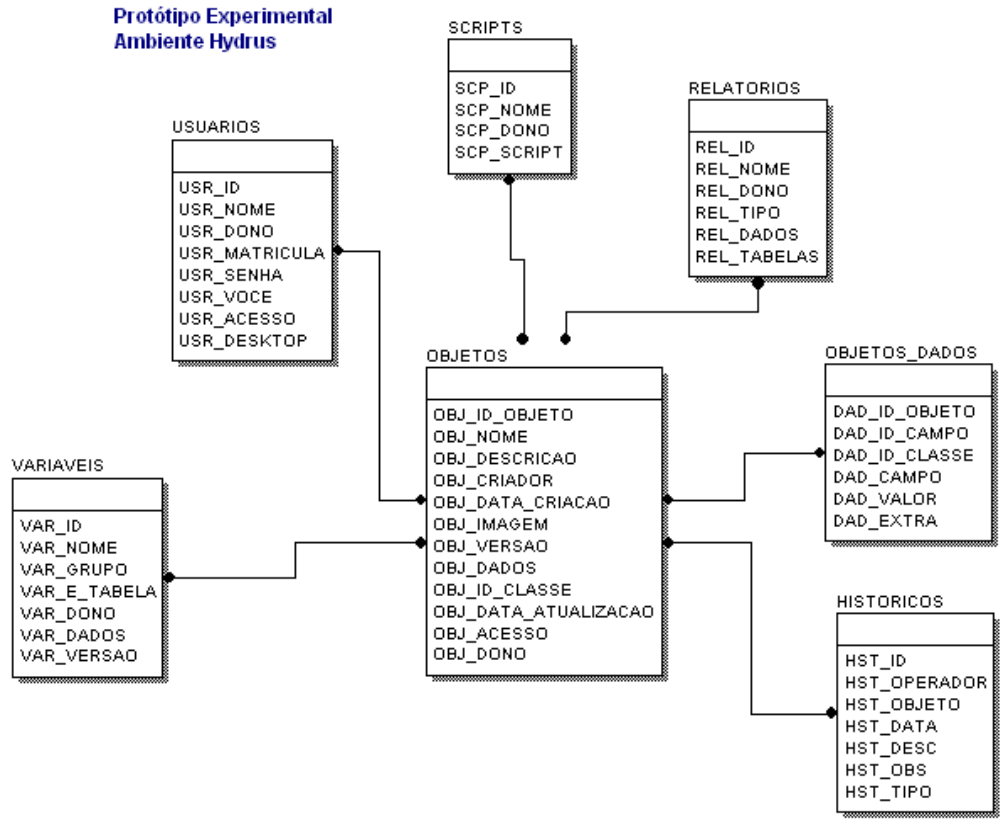
O repositório de metadados foi criado através de uma base de dados Paradox. A persistência dos objetos foi feita usando a técnica de mapeamento dos atributos para campos nas tabelas. Cada classe persistente do protótipo ganhou uma tabela específica para os seus atributos exclusivos. Para armazenar os atributos comuns a

todas as classes foi definida a tabela “objetos”. Cada tabela específica se relaciona com a tabela “objetos” através do Id do objeto. Na figura 25 está apresentado o esquema ER do banco.

Duas novas tabelas foram incorporadas ao banco de dados: “objeto_dados” e “histórico”. A primeira foi utilizada para armazenamento de estruturas complexas, enquanto a segunda, os históricos das operações sobre os objetos do protótipo.

Embora a classe clsLista tenha sido definida como descendente de clsObjetoPersistente, no protótipo, para simplificar uma série de operações, isto não aconteceu e ela foi implementada como uma classe independente, ganhando uma tabela especial. No entanto, todos os métodos e aspectos definidos no ambiente Hydrus em relação a ela foram mantidos.

Figura 25 - Esquema ER das classes persistentes



Fonte: o autor (2004)

5.5.4 Acesso aos dados

No protótipo, o acesso aos dados foi realizado usando diretamente os componentes nativos do Delphi, o TDataBase, TTable e TQuery ao invés de clsConexao e clsDataset. Esta decisão foi tomada porque:

- O uso direto dos componentes de acesso a dados do Delphi nos scripts do protótipo era possível, o que dispensava a criação das classes como pontes para estes componentes;
- Todo o gerenciamento de listas de conexões e datasets já estava bem implementado na linguagem Object Pascal e, embora possível, haveria um investimento razoável em desenvolvimento para reimplementá-lo.
- As classes clsConexao e clsDataset, especificamente neste protótipo, serviriam apenas como fachada, criando um overhead desnecessário.

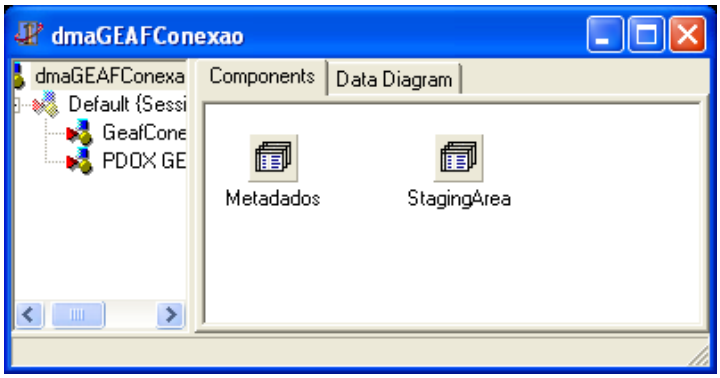
Apesar das justificativas anteriores, o uso direto dos componentes de acesso a dados do Delphi transferiu para os scripts a incumbência de persistir as definições de conexões e acesso aos dados das bases. Foram mudanças que não seriam realizadas se a implementação fosse destinada a produção ao invés de teste do ambiente.

5.5.5 Sessões e controle do ambiente

O conceito de sessão no protótipo foi simplificado. Nele existe apenas uma sessão aberta automaticamente na carga do ambiente. As conexões com o repositório de metadados e área temporária são estabelecidas na abertura da sessão e são definidas através de dois componentes Tdatabase mostrados na figura 26

.

Figura 26 - Conexões com o repositório central de metadados e Staging Área



Fonte: o autor (2004)

Com a inexistência de múltiplas sessões, dispensou-se, no protótipo, a implementação das funcionalidades relacionadas ao arquivo de configuração e todas as operações referentes ao gerenciamento de sessões.

A carga de qualquer objeto persistido no ambiente Hydrus é feita através da classe clsSessao. No protótipo, o processo foi facilitado através da carga automática de objetos descendentes de Tobjeto via passagem de Id. Qualquer objeto do repositório sabe como ler os seus valores do repositório de metadados. Então, para carregá-lo simplesmente lhe forneça o seu Id, como mostrado na figura 27.

Figura 27 - Criação Objeto a partir do seu Id

```
// Cria a instância referente ao script 1234
varScript := TElsScript.create(nil, 1234 {Id});
```

Fonte: o autor (2004)

5.5.6 Script

Todas as funcionalidades definidas para o ambiente Hydrus em relação aos scripts foram implementadas no protótipo. Inclusive foi dada atenção especial à interface que ganhou sintaxe colorida, editor de formulários e facilidades para edição como pesquisa, copiar, colar.

Usando-se componentes de terceiros, pertencentes ao pacote da Dream Company, foi possível capacitar os scripts com acesso total a todos os elementos da linguagem de Object Pascal. Isto potencializou a implementação de códigos e permitiu o uso de importantes conjuntos de classes nativas do Delphi, na programação dos

scripts. Além disto, todos os objetos guardados no metadados foram disponibilizados também nos scripts, permitindo a extensão do ambiente através deles.

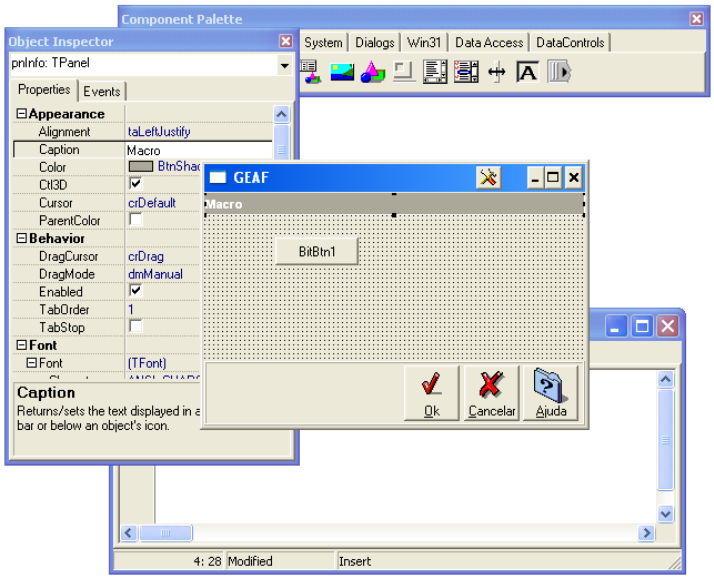
Os scripts do protótipo usam o Object Pascal como linguagem de programação. Além disto, eles possuem um analisador de sintaxe que além de apresentar os erros em caso de falha, ainda posicionam o programador no ponto onde o problema aconteceu.

Uma das características que dará maior produtividade aos programadores de scripts do protótipo é o editor de formulário. Ele é flexível e permite a criação de janelas com o profissionalismo encontrado nos grandes ambientes de desenvolvimento. A figura 28 apresenta a edição simples de uma janela. Em conformidade com a proposta para o ambiente Hydrus, cada script no protótipo pode estar associado a vários formulários que são usados durante a sua execução. Além disto, os formulários podem ser acessados e modificados de dentro do código dos scripts, fato que dá plenos poderes ao programador para escolher a maneira como deseja trabalhar com as janelas dos seus programas.

Após a conclusão do protótipo, foi simulada a sua utilização em um ambiente de produção. A conclusão chegada foi que o usuário final precisa de um caminho simples e familiar para alcançar e executar os scripts do ambiente. Não adiantou disponibilizar os scripts organizadamente em uma estrutura hierárquica de árvore, pois o usuário não aprovou a disposição. Ele sentiu dificuldade em utilizá-los. A solução dada à questão foi melhorar a interface do protótipo e modificar a forma como os scripts foram disponibilizados, criando-se:

- Menus configuráveis que são definidos em tempo de execução e que são associados aos scripts. Quando o usuário aciona um item destes menus, é executado o script ligado ao item.
- Um ambiente WEB onde o usuário pode navegar através de páginas HTML. Neste contexto, alguns links das páginas são associados aos scripts do protótipo. Quando um link é acionado, o script associado é executado.

Figura 28 - Editor de formulário



Fonte: o autor (2004)

5.5.7 Plugins

Os plugins não foram implementados no protótipo sendo deixados para futuros trabalhos. Infelizmente a ausência deles enfraqueceu o experimento e serviu para mensurar, empiricamente, o quanto eles são essenciais para o ambiente Hydrus. Embora todos os problemas formulados puderam ser resolvidos através dos scripts, faltou a segurança e velocidade que só os plugins proporcionam.

Para minimizar os efeitos causados pela ausência de plugins na ferramenta experimental, foi definida uma classe especial usada na simulação de uma biblioteca de funções incorporada aos scripts. Nesta classe foram criados métodos que forneceram aos scripts funcionalidades básicas e úteis como manuseio de planilhas Excel, movimentação de dados entre tabelas, e funções gerais de data e moeda.

5.5.8 Segurança

Devido à simplicidade do protótipo não justificou a criação de um controle de acesso completo com todas as funcionalidades sugeridas nas especificações do ambiente Hydrus. Apesar disto, características como grupo de acesso, permissões, usuários e suas senhas, foram implementadas, embora com limitações. Por exemplo,

o número de grupos foi restringido a oito e as permissões ou privilégios foram definidos apenas ao nível de menus e objetos.

Testou-se o controle de acesso ao nível de objetos através do sistema de segurança do protótipo. A conclusão obtida foi que o usuário deseja restrição de acesso a um nível maior, como item de um menu, ao invés de restrições pontuais, como nos objetos.

5.5.9 Utilitários e Configuração

Os utilitários, ao contrário de Java que são criados ao nível de objetos, são implementados em Delphi como funções. No protótipo, foi programado um conjunto deles, organizados em “units”.

Os aspectos definidos para o ambiente Hydrus em relação a configurações foram implementados no protótipo. Inclusive duas configurações importantes mencionadas anteriormente foram definidas:

- Configuração do menu principal, associando seus itens aos scripts;
- Definição de páginas HTML, associando os seus links aos scripts;

6 CONSIDERAÇÕES FINAIS

6.1 CONSIDERAÇÕES GERAIS

Como foi observado, existem muitas soluções para o problema de povoamento de Data Warehouse, cada uma atendendo a um conjunto de aspectos do processo. As organizações optam por aquelas que melhor atendem a suas necessidades, analisando os custos e os benefícios que elas apresentam. No entanto, uma solução completa, que seja adequada a todas as situações ainda não existe. Por isto o ambiente Hydrus surge como uma opção viável devido a sua flexibilidade e capacidade de se ajustar aos diversos tipos de problemas, mesmo sendo uma solução mais simples

O processo ETL, pela sua complexidade e variedade de situações, não será resolvido simplesmente utilizando soluções baseadas em ferramentas complicadas e com inúmeras funcionalidades, que muitas vezes são difíceis de serem utilizadas, ou geram um custo referente a treinamento e preparação que ultrapassa o limite estipulado pelas adquirentes. Além disto, as organizações precisam de resultados imediatos e soluções ajustadas às suas realidades, que sejam simples, mas que atendam aos requisitos específicos de seus projetos de DW.

Tendo uma visão bem definida de como os problemas ETL se apresentam e a forma como as soluções são criadas, o Hydrus foi concebido visando oferecer um ambiente onde o problema de povoamento de DW fosse resolvido através de um conjunto de artefatos que são gerados pelas organizações, durante a implementação de seus projetos, e compartilhados ou reaproveitados em outras ocasiões. A arquitetura voltada a extensão fornece um ambiente capaz de atender a vários tipos de situações, onde ferramentas ETL, com características variadas, podem ser implementadas, independente da sua complexidade.

A ideia de um repositório centralizado, que armazena tanto a ferramenta como o processo que ela implementa, faz do Hydrus um ambiente diferenciado. Analisando as soluções disponíveis, percebeu-se que as ferramentas sempre são fornecidas como pacotes fechados, e o desenvolvimento de uma solução ETL é feito a partir das estruturas que elas fornecem. Por isto, muitos projetos ETL são implementados utilizando-se várias ferramentas, em conjunto, porque os requisitos destes projetos não conseguem ser inteiramente atendidos apenas por uma única. Este quadro

forneceu a motivação necessária à concepção do ambiente Hydrus, pois se concluiu, através dos estudos realizados, que o desenvolvimento do processo de povoamento dos DW deveria ser feito ao nível do processo e do ferramental.

Aspectos fundamentais do ambiente Hydrus foram empregados em projetos práticos, e mostraram-se eficazes na solução de problemas onde a capacidade de extensão e ajustes dinâmicos do ambiente da aplicação era requerida. Os bons resultados obtidos nos sistemas que serviram como testes, com respostas adequadas e efetivas às situações variadas apresentadas, mostrou que as características definidas para o Hydrus são funcionais e aplicáveis aos problemas reais.

Embora já tenham sido aplicados, na prática, conceitos definidos no ambiente Hydrus, ele ainda deve ser considerado experimental. Implementações do ambiente, como o protótipo descrito no capítulo anterior, poderiam ser utilizadas por projetos de pesquisa que tratem de aspectos relacionados ao processo ETL.

6.2 CONTRIBUIÇÕES

A pesquisa apresentada nesta dissertação contribuiu em diversas áreas referentes ao processo de povoamento de Data Warehouse, que são citadas a seguir:

- Um estudo detalhado sobre os aspectos ligados ao povoamento de Data Warehouse foi apresentado, inicialmente de forma abrangente ao serem mostradas as principais características do DW, e, especificamente, quando foram detalhadas as etapas do processo ETL.
- A análise realizada sobre a adoção de ferramentas pelas organizações versus o desenvolvimento de soluções internas, ressaltando os principais aspectos a serem considerados durante o processo de avaliação do ferramental, e conseguinte seleção. O estudo abordou de forma imparcial as características do processo de escolha de ferramentas ETL, e apresentou um roteiro que pode ser usado na sistematização da seleção.
- O levantamento de ferramentas ETL consagradas, que são utilizadas atualmente pelas organizações, e que apresentam características diferenciadas uma das outras, como geração de código e ambiente de desenvolvimento. Os aspectos mais importantes destas ferramentas foram apresentados, e eles servem como ponto de partida para uma pesquisa mais detalhada e abrangente sobre o assunto.

- Uma proposta simples para persistência de objetos a partir da representação texto de suas instâncias foi apresentada no penúltimo capítulo. Nesta proposta foi sugerida a utilização de XML, ou de formato específico, relacionado com a linguagem de desenvolvimento adotada na implementação das classes.
- Finalmente, a maior contribuição apresentada foi o ambiente Hydrus, com sua arquitetura e componentes, que apresenta uma proposta real capaz de apoiar a construção de ferramentas ETL e seus processos.

6.3 TRABALHOS FUTUROS

Vários aspectos do processo ETL foram abordados neste trabalho e contemplados na definição do ambiente Hydrus. No entanto, alguns deles foram tratados de forma geral e podem ser mais aprofundados, para que a concepção do ambiente arquitetado seja mais completa e detalhada, e para permitir uma implementação mais padronizada nas diversas plataformas e linguagens.

Plugin é um dos pilares que sustenta a capacidade de extensão do ambiente Hydrus. Embora tratado de forma adequada no texto desta dissertação, sugere-se que seja ampliado o estudo sobre o gerenciamento deles, de forma que os vários aspectos sobre o assunto sejam mais bem investigados, e um modelo geral para a incorporação, armazenamento, utilização e compartilhamento destes artefatos seja definido. Existem, atualmente, alguns modelos práticos de gerenciamento de plugins disponíveis, tal como o definido pela IBM para a plataforma Eclipse (IBM, 2003) . Este modelo pode servir de base para um estudo mais detalhado, que pode ser adotado pelo ambiente Hydrus. Embora o Eclipse seja totalmente baseado em plugins, esta plataforma ainda não apresenta uma solução adequada para o gerenciamento corporativo deles.

A solução apresentada para persistência de objetos e entidades do ambiente Hydrus foi satisfatória para o nível de detalhamento desejado para o assunto neste trabalho. Mas uma interessante pesquisa nesta área poderia ser realizada, onde seriam abordados temas como adoção de linguagens universais, como XML, para a representação dos objetos do repositório. A forma como o compartilhamento entre diversas sessões ou ferramentas poderia ser implementada no ambiente Hydrus.

A implementação do ambiente Hydrus foi realizada ao nível de protótipo, onde apenas os aspectos mais relevantes foram programados. Como um interessante trabalho, sugere-se a implementação do ambiente com todos as suas características, em uma linguagem de desenvolvimento diferente da adotada no protótipo. O trabalho poderia ser bem realizado sobre a plataforma Eclipse, por exemplo, onde a arquitetura direcionada a plugins poderia ser utilizada para minimizar os esforços de programação, e fornecer mais robustez a versão do ambiente a ser criado. Além disto, como a linguagem natural da plataforma Eclipse é Java, a implementação do ambiente Hydrus ganharia escala e portabilidade.

REFERÊNCIAS

ADELMAN, Sid & OATES, Joe **Data Warehouse Project Management** – DM Review, May., 1998.

ARANGO, G., Prieto, R. **Domain Analysis Concepts and Research Directions In "Domain Analysis and Software System Modeling"**, 1st ed., California, IEEE Computer Society Press Tutorial, pg. 09-25, 1991.

BALLARD, Chuck & HERREMAN, Dirk **Data Modeling Techniques for Data Warehousing** – IBM, International Technical Support Organization, Feb., 1998.

BALLARD, Chuck; HERREMAN, Dirk; SCHAU, Don; BELL, Rhonda; KIM, Eunsaeng; VALENCIC, Ann. "**Data Modeling Techniques for Data Warehousing**". International Technical Support Organization, Feb., 1998.

BARQUINI, RAMON - **Planning and designing the Warehouse**, New Jersey, Prentice-Hall,1996, 311 pg.

BATINI, C. e LENZERINI, M. - **Comparative Analysis of Methodologies for Database Schema Integration**, ACM Computing Surveys, New York, v.18, nº 4, pg.323-364, Dez., 1986.

BOEHM, B., SCHERLIS, W. **Mega Programming**. Proceedings of the DARPA Software Technology Conference, Arlington, VA, Apr., 1992.

BOKUN, M., TAGLIENTI, C. **Incremental Data Warehouse Updates,** obtido em: http://www.dmreview.com/dmreview/, consultado em 2000.

BRACKETT, M. **The DataWarehouse Challenge - Taming Data Chaos**. John Willey & Sons, Inc, 1996, 579p.

BRAGA R., WERNER, C., MATTOSO, M. **Odyssey: A Reuse Environment based on Domain Analysis**. Proceedings of the IEEE Symposium on Application - Specific Systems and Software Engineering Technology, IEEE CS Press, pg.50-57, Richardson, Texas, Mar., 1999.

BUSCHMANN, F. et al. **Pattern-Oriented Software Architecture: A System of Patterns**. John Wiley & Sons, 1996.

CAMPOS, Maria Luiza, FILHO, A.V. Rocha. **Data Warehouse**. In: XVII Congresso da Sociedade Brasileira de Computação – XVI Jornada de Atualização em Informática. Brasília, 1997, pg 221-261.

CHAUDHURI, S. e DAYAL, U. **An Overview of Data Warehousing and Olap Technology**, SIGMOD Record, New York, v.26, nº 1, pg.65-74, Mar., 1997.

CHEN, Peter Pin-Shan; "**The Entity-Relationship Model: Toward a Unified View of Data**". ACM Transactions on Database Systems, Vol1, Num1, pp 9 – 36, Jan., 1976.

CODD, E. F. Twelve **Rules for Online Analytical Processing**, Computerworld, Abr., 1995.

COLLIN, J. White, **An Analysis-led Approach to Data Warehouse Design and Development** – Article from Database Associates, sponsored by Evoke Software – 2001

COSTA, André Fernandes e Anciães, FELIPE Curvello, **Aspectos De Criação E Carga De Um Ambiente De Data Warehouse**, Rio de Janeiro, Mar., 2001.

CRAIG, R., VIVONA, J., BERCOVITCH, D. **Microsoft Data Warehousing – Building Distributed decision** 133 support Systems. Wiley Computer Publishing, 1999. 368 p.

DATA HABITAT CORPORATION, **DataHabitat ETL User Guide**, 2003. 54 p.

FABRIS, Carem Cristina, **Um Estudo Sobre Data Warehouse**, jan. 2000, 74p.

FAISAL Shah, Knightsbridge Solutions, ETL: **Pay Now or Pay Later**, Nov., 2003, obtido em http://www.dw-institute.com/research/display.asp?id=6874, consultado em 2004.

FAYAD, M., JOHNSON, R. **Domain-Specific Application Frameworks: frameworks experiences by industry**. John-Wiley & Sons, New York, NY, 1999.

FAYYAD, Usama M. **Advances in Knowledge Discovery and Data Mining**, AAAI Press – Los Angeles - USA, 1996.

FISCHER, G. **Domain-Oriented Design Environments. Automated Software Engineering** – The International Journal of Automated Reasoning and Artificial Intelligence in Software Engineering, Vol.1, No.2, pg177-203, Jun., 1994.

FURNAS, G.; **Generalized fisheye views**. Proceedings CHI'86, 1986, 16- 23.

GOLFARELLI, M.; MAIO, D.; RIZZI, S.; "**The Dimensional Fact Model: A Conceptual Model for Data Warehouses**". Int'l J. Cooperative Information Systems (IJCIS), vol. 7, n. 2-3, 1998.

GOODYEAR, Mark, RYAN, Hugh W., SARGENT, Scott R., et al. **Netcentric and Client/Server Computing – a practical guide**. Andersen Consulting. Boca Raton: Auerbach, 1999.

Grupo DATAAWARE, obtido em http://genesis.nce.ufrj.br/dataware/, consultado em 2003

HACKATHORN, R. – **Data Warehousing Energizes Your Enterprise**. Datamation, New York, v.41, nº 02, pg.38-43, Fev., 1997.

IBM, Object Technology International, **Eclipse Platform Technical Overview**, Feb., 2003, obtido em http://www.eclipse.org/, consultado em 2004

IDG, Revista de computação, obtido em http://www.lidgnow.com.br, consultado em 1998 - Edição 237 – Out., 1998.

INMOM, W. H., ZACHMAN, J. A., GEIGER, J. G. **Data Stores Data Warehousing and The Zachman Framework Managing Enterprise Knowledge**. McGraw-Hill, 1997. 358 p.

INMON, W. H. **Building The Data Warehouse**. John Wiley & Sons, Inc., 1992. 298p. Enterprise Meta Data. obtido em http://www.dmreview.com/portal_ros.cfm?NavID=114&Type=1&opic=11&PortalID=18, consultado em 2003

INMON, W. H., HACKATHORN, R. **Using The Data Warehouse**. John Wiley & Sons, Inc., 1994. 285 p.

INMON, W.H. – **Building the Data Warehouse**, John Wiley & Sons Inc., New York, 1996.

INMON, W.H. **Como Usar o Data Warehouse**, Info book, Rio de Janeiro, 1997.

KACHUR, Richard, **The Data Warehouse Diary: Initiating a Data Warehouse Project**, Part I – DM Review, Dec., 2000

KELLY, S. **Data Warehousing in Action**. New York, John Wiley & Sons, 1997.

KIMBALL, R., REEVES, L., ROSS, M., THORNTHWAITE, W. **The Data Warehouse Lifecycle Toolkit – Expert Methods for Designing, Developing and Deploying Data Warehouse**. New York: John Wiley & Sons, Inc., 1998. 771p.

KIMBALL, Ralph – **The Data Warehouse Toolkit**, John Wiley & Sons Inc., New York, 1996.

LANG, Roger, 5 **Hardware's Considerations** – Bank Marketing, v.29, n.4, Feb., 1997, p.26

MARAKAS, George M. – **Decision Support Systems in the 21st century, Prentice** - Hall Inc., USA, 1999.

MCDOWELL, R. C., CASSEL, K.A. **The RLF Librarian: A Reusability Librarian based on Cooperating Knowledge-Based Systems**. Proceedings of the 4th Annual Rome Air Development Center Knowledge-Based Software Assistant Conference, Utica, NY, Sep., 1989.

MELO, Rubens Nascimento. **Data Warehousing (Tutorial)**. In: XIII SBBD – Simpósio Brasileiro de Banco de Dados, Salvador, 1997.

MICROSOFT Corporation. **Microsoft SQL Server 7.0 Data Warehousing Framework**. 1998, obtido em http://www.microsoft.com/sql/70/gen/whatsnew.htm, consultado em 1998.

MICROSOFT Corporation. **Windows DCOM Architecture**. 1998, obtido em http://www.microsoft.com/oledev/olecom, consultado em 1998

MORAES, R. L. – **Sistemas de Data Warehouse**, obtido em http://www.inf.ufrgs.br/ ~rlmoraes/dw.html, consultado em 2002.

MORAES, R.L. **Data Warehouse**. Trabalho Individual, Curso de Mestrado em Ciência da Computação, UFRGS, RS, 1997.

NEIGHBORS, J. M. **Software Construction from Components**. PhD Thesis, TR-160, ICS Department, University of California at Irvine, USA, 1980.

PEREIRA, Denise Marciel, Uso do Padrão OIM de Metadados no Suporte às Transformações em Ambiente de Data Warehouse, Rio de Janeiro, 2000, obtido em http://genesis.nce.ufrj.br/dataware/Metadados/Teses/Denise/pagina_tese.htm, consultado em 2000

RAVAT, F., Teste, O., ZURFLUH, G. **Towards Data Warehouse Design**. Proceedings of the International Conference on Information and Knowledge Management (CIKM), Missouri, USA, 1999.

SANTOS, Roberto Ângelo Fernandes, **Metodologia E Uso De Técnicas De Exploração E Análise De Dados Na Construção De Data Darehouse**, Recife 2002.

SCOTT, W. Ambler, **The Fundamentals of Mapping Objects to Relational Databases**, 2003-2004, obtido em http://www.agiledata.org/essays/mappingObjects.html, consultado em 2004

SILVERSTON, L., INMON, W.H., GRAZIANO, K. **The Data Model Resource Book**. John Wiley & Sons, New York, NY, 1997.

SINGH, Harry, **Data Warehousing: Concepts, Technologies, Implementations, and Management** – Upper Saddle River, NJ: Prentice Hall, 1997

STAUDT, M., VADUVA, A., VETTERLI, T. **The Role of Metadata for DataWarehousing**. Institut für Informatik der Universität Zürich Technical Report, ifi-99-06, 1999.

STEVEN R. Meyer, **Which ETL Tool is Right for You?** Published in DM Direct in Jun., 2001

SWEIGER, Mark; MADSEN, Mark R.; LANGSTON, Jimmy; LOMBARD, Howard; Clickstream Data Warehousing, 1st edition, Jan., 2002

TRONCHIN, Valsoir, **Análise, Modelagem e Implementação de Data Warehouse** – São Paulo: Fenasoft98, 1998

VASSILIADIS, P. **Gulliver in the Land of Data Warehouse: Practical Experiences and Observations of a Researcher**. Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW2000), Stockholm, Sweden, Jun., 2000.

WILLIAMS, J. – Tools for Traveling Data. DBMS, California, Jun., 1997

YOUNESS, S. **Professional Data Warehousiong with SQL Server 7.0 and OLAP Services**. Wrox Press Ltd, 2000. 604 p.

ANEXO A – AMBIENTE REDIRIS

O conteúdo deste anexo compreende o capítulo dois da tese de mestrado de Roberto Ângelo Fernandes Santos com o título “Metodologia e Uso de Técnicas de Exploração e Análise de Dados na Construção de Data Warehouse”, e serve como material elucidativo sobre o ambiente Rediris, em cujo contexto o ambiente Hydrus pode se encaixar e agregar valor.

A.1 O AMBIENTE REDIRIS

Os projetistas de Data Warehouse precisam utilizar-se de técnicas apropriadas para desenvolvimento e implantação de Sistemas de Informação, porque as organizações demandam cada vez mais informações de qualidade e cada vez mais rápido. Essa demanda só pode ser suprida por um ambiente voltado para esse fim, construído e/ou adaptado em tempo hábil utilizando sempre o conhecimento adquirido em implementações anteriores. Nesse contexto que surge a idéia de um ambiente integrado chamado de REDIRIS (*Reuse Environment on Data Integration, Reuse and Quality in Information Systems*). Neste capítulo será apresentado o ambiente REDIRIS de forma simplificada, para contextualizar e situar os mecanismos desenvolvidos neste trabalho.

A.2 INTRODUÇÃO

O desenvolvimento do SAD, encorajado por todo o avanço no campo do Data Warehouse, vem tornando-se uma motivação comum de pesquisadores na área (VASSILIADIS, 2000). Contudo, a notória complexidade relacionada a construção de um esquema dimensional eficiente, adicionou uma considerável quantidade de projetos malsucedidos (KELLY, 1997) (SILVERSTON, INMON, GRAZIANO, 1997) reivindicando por uma reflexão mais profunda nas estratégias adotadas no desenvolvimento. Alguns fatores relevantes como o pouco conhecimento do negócio, necessidade do limite de gerência, abordagens de projeto inadequadas, fraca qualidade dos dados e falta do conhecimento dos dados, aparecem sendo uma causa comum das possíveis falhas. Assim, o reuso de modelos e soluções, cuja eficiência já tem sido reconhecida, leva a uma perspectiva promissora para sistemas de apoio à

decisão, baseados na premissa que estes sistemas dividem um conjunto de conceitos em comum (KIMBALL, 1996). Tal abordagem envolve a identificação dos objetos, operadores e relações que compõem um domínio de aplicação (NEIGHBORS, 1980) (ou de uma família de aplicações) como meios de gerar novos exemplares de soluções pré-existentes. Na Engenharia do Domínio (ARANGO, 1991) técnicas e métodos suportam inteiramente o processo de capturar, organizar e melhorar a informação com respeito a um domínio SAD particular, enquanto encapsula a dado coletado em componentes reusáveis e modelos. Mais recentemente, outras tecnologias floresceram no suporte de reuso do domínio específico tal como o metadado (STAUDT, VADUVA, VETTERLI, 1999), reuso de bibliotecas (MCDOWELL, CASSEL, 1989), mega programação (BOEHM, SCHERLIS, 1992), modelagem orientada a aspectos, testes padrões (BUSCHMANN, 1996), e estruturas de frameworks (FAYAD, JOHNSON, 1999). Apesar da atrativa perspectiva, o desenvolvimento de apoio à decisão baseado no reuso não tem feito parte das mais recentes pesquisas nesta área, de acordo com o exame descrito em (VASSILIADIS, 2000).

Por outro lado, muitos problemas normalmente encarados no último estágio de um projeto de SAD poderiam ter sido realizados em etapas adiantadas da construção se a investigação apropriada dos dados fosse conduzida. De fato, as investigações de dados induzem projetistas a considerarem meticulosamente a qualidade de dados em todos os níveis de decisão e contribuem para uma compreensão melhor da organização e do potencial do projeto. Técnicas de inteligência artificial, tais como a mineração de dados (FAYYAD, 1996), ajudam na descoberta da informação latente que eventualmente reside nos dados operacionais, além de auxiliar nas etapas de projeto e implementação de um SAD. Em um SAD, a complexidade de cada análise pode ser consideravelmente reduzida se nós dividirmos os dados em unidades menores (de acordo com similaridades de negócio) onde cada amostra de dados mais restrita pode ser extraída e analisada. Esta abordagem também está de acordo com princípio do *Data Mart,* defendido por muitos autores (KIMBALL, 1996) (INMON ,1997) (MARAKAS, 1999).

REDIRIS guarda uma grande similaridade com Ambientes de Projeto Orientados a Domínio - Domain-Oriented Design Environments (DODE) - (FISCHER, 1994) como uma Odisséia – Odyssey- (BRAGA, WERNER, MATTOSO, 1999) estendendo à automatização do reuso das soluções sob a arquitetura baseada em componente. REDIRIS considera o processo de desenvolvimento reusável, o qual aponta para a

redução da complexidade inerente à definição de soluções e assim construir em um tempo menor e com mais qualidade aplicações de suporte a decisão.

A.3 CONCEITOS BÁSICOS

Alguns princípios básicos regem a concepção e a construção do ambiente REDIRIS. A seguir são apresentados esses princípios.

- *Primeiro Princípio*. Qualidade como o maior fator de sucesso, desde os processos de captura e análise dos dados, até a entrega de uma nova solução de apoio à decisão.
- *Segundo Princípio*. O processo de desenvolvimento SAD clássico é inadequado. Consequentemente, incorporação do conhecimento de negócio e exigências de gerência em estágios avançados deste processo, aliados ao uso de métodos de desenvolvimento avançado contribuem para impedir que o processo total falhe.
- *Terceiro Princípio*. A dinâmica de um SAD requer velocidade, respostas confiáveis, que pedem a implementação de um esquema de reuso eficiente, assim como ela encaixa processos de decisão, acoplando modelos (semi) automáticos de métodos de geração e ferramentas inteligentes.
- *Quarto Princípio*. O ciclo de desenvolvimento de um SAD deve perseguir a interatividade, características dinâmicas e incrementais, e requer avaliação durante todos os estágios do processo, com uma atenção especial para as mudanças estratégicas que podem ocorrer dentro da organização.
- *Quinto Princípio*. A especificação de metadados e domínios de aplicação, aliados com a sua gerência eficiente remanesce os grandes desafios de um ambiente de construção de um SAD.

Acoplado com os princípios antigos, REDIRIS introduz técnicas de inteligência artificial ao ciclo de desenvolvimento de um SAD como um caminho para conseguir objetivos na qualidade de dados. Entre outros aspectos, essas técnicas relacionadas a aplicações de indução de regras para descobrimento e avaliação do conhecimento de negócio, casamento e correlação de algoritmos (matching) para tratar anomalias de dados, algoritmos de associação para identificação de hierarquias dimensionais, e

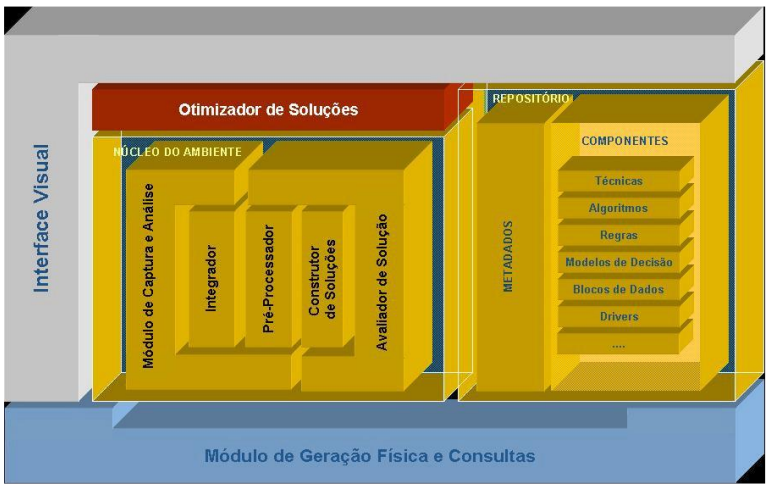
métodos estáticos para desenvolvimento de soluções de apoio à decisão de alta qualidade.

Considerando o problema da integração de dados, não se pode ignorar a *World Wide Web* como um principal recurso de dados prontos e disponíveis, mas de forma muito heterogênea. A semântica da integração dos metadados, em conjunto com semânticas ontológicas formarão a base dessa futura arquitetura. O REDIRIS foca no desenvolvimento do processo semiautomático para semântica de integração do esquema, baseado em padrões XML e RDF.

A.4 A DINÂMICA REDIRIS

A arquitetura REDIRIS é descrita na Figura 29. O componente de captura e análise responde para os dois processos sobre a captura de aspectos herdados de uma aplicação de domínio, e a análise de sistemas legados para avaliar seus potenciais para reuso e qualidade de dados operacionais. A respeito da funcionalidade anterior, o componente de Captura tem sido designado para coletar informações reais e dimensionais deixando para trás índices estratégicos organizacionais, corretamente traduzidos em expressões de domínio específico conhecido.

Figura 29 - Arquitetura do REDIRIS



Fonte: o autor (2004)

A integração de Dados é um dos mais importantes requerimentos do Pré-Processador. Em alguns casos, quando os dados residem em distribuição, recursos heterogêneos, o componente Integrador fornece o auxílio em recuperar o construtor

dos dados de cada arquivo independente ou bases compartilhadas, dando à informação recolhida um formato original ao ambiente como prescrito nas regras de semântica integrativa. Além disso, o Integrador ajuda no trabalho de análise de dados gerando tabelas de exemplos concordando, com uma predefinida granularidade de dados. O Pré-Processador executa a tarefa importante de promover técnicas de análises de dados previamente descritas para ajudar na avaliação de seu potencial de apoio à decisão interno. Além disso, protótipos criados, assim como soluções pré-existentes podem ser carregadas no repositório REDIRIS, para enriquecer e propiciar o reuso de componentes de domínio semelhantes em novas soluções. A função de construção de novas soluções a partir do zero ou baseadas em soluções antigas é uma funcionalidade do componente Construtor. Este componente tem a sua disposição a biblioteca baseada em reuso do REDIRIS para obter interativamente os componentes de apoio à decisão com as características da nova solução. O conhecimento da aprendizagem adquirido durante este processo de construção é retroalimentado ao metamodelo do ambiente. O Construtor também interage com o componente Otimizador para ajudar usuários na seleção do melhor conjunto de configuração para um determinado modelo multidimensional, baseado em regras de configuração encaixadas, ativado em conformidade com as características especificadas da solução. Finalmente, todos os componentes REDIRIS acima mencionados são controlados por uma iterativa Interface multifuncional que tem o desafio de personalizar todo o processo interativo para diferentes perfis de usuários, acesso inteligente aos componentes de domínio, integração de unidades de ambiente, e recuperação eficiente de dados em diferentes níveis abstratos.

A.5 OUTROS AMBIENTES RELACIONADOS

REDIRIS guarda uma grande similaridade com Ambientes de Projeto Orientados a Domínio - *Domain-Oriented Design Environments* (DODE) - (FISCHER, 1994) como uma Odisséia – *Odyssey* - (BRAGA, WERNER, MATTOSO, 1999) à extensão de automatizar reusar das soluções sob a arquitetura baseada em componente. Contudo, a que interesses o domínio específico de sistemas de apoio à decisão, REDIRIS representa um detalhamento maior, abordagem sustenta em suporte de um processo de desenvolvimento reusável, o qual aponta para a redução da complexidade inerente a definir tais soluções e assim entregar em um tempo menor com mais qualidade aplicações de apoio à decisão.

Similarmente, o protótipo WarGen (RAVAT, ZURFLUH, 1999) provê uma plataforma eficiente para geração de aplicações Data Warehouse baseada em soluções da gerência da base de dados orientada a objeto. Uma interface gráfica e um gerador automatizado permitem aos usuários especificar interativamente e incrementalmente novas soluções que podem ser criadas usando classes de objetos genéricos. REDIRIS estende essa ideia para disponibilizar um ambiente completo não somente focado na geração de soluções de um SAD (como *Data Warehouse*), mas também com o objetivo de melhorar soluções existentes e fazendo experiências reusáveis durante todo o processo de geração.

A.6 O COMPONENTE PRÉ-PROCESSADOR NO REDIRIS

Para essa dissertação o módulo de pré-processamento é o mais importante, porque esse trabalho tem no seu centro uma metodologia focada na manipulação de dados para construção de Data Warehouses. Apesar desse módulo ser o mais importante, os outros módulos também são requeridos para a elaboração da dissertação. No Capítulo 5 da tese é mostrado um ciclo por todos os componentes do REDIRIS envolvidos, focando principalmente a importância nesse trabalho.

O módulo PRÉ-PROCESSADOR do ambiente REDIRIS é o responsável por toda a captura de metadados a partir dos dados de entrada. Ele é quem faz toda e qualquer manipulação de dados através de análise, limpeza e transformação dos dados. Ele executa operações relacionadas com os dados de entrada; fornece metadados para a modelagem dimensional, verifica a qualidade e enriquece os dados de entrada.

A análise dos dados é feita durante todas as etapas que envolvem dados no REDIRIS, porém é o módulo pré-processador que gera a maior parte dos metadados de suporte a essa operação. Ele também provê acesso às técnicas e algoritmos de limpeza e transformação de dados. A limpeza e transformação dos dados envolvem a verificação da consistência das informações, a correção de possíveis erros, o preenchimento ou a eliminação de valores nulos e redundantes e a geração de novos atributos derivados.

O pré-processador é composto basicamente por um conjunto de processos, que manipulam e geram metadados a partir de uma amostra de dados. Esses processos utilizam uma extensa biblioteca de técnicas e algoritmos de tratamento de dados,

localizados no Repositório do REDIRIS. Todas essas técnicas e algoritmos são executados a partir dos metadados que as representam, o que possibilita a inserção de novas técnicas, tornando o pré-processador um módulo extensível a qualquer nova implementação requerida.