Departamento de **C**iência da **C**omputação
Instituto de **M**atemática e **E**statística
Universidade de **S**ão **P**aulo

Trabalho de Conclusão de Curso

Atualização Incremental De Data Warehouses

Pedro Paulo de Souza Bento da Silva
<pedrosbs@linux.ime.usp.br>

Orientador: João Eduardo Ferreira
<jef@ime.usp.br>

Co-Orientador: Pedro Losco Takecian
<plt@ime.usp.br>

São Paulo, 1º de dezembro de 2009

Sumário

1 Resumo

Um Data Warehouse (DW) é uma estrutura de tabelas que integra um grande volume de dados proveniente de bancos de dados transacionais não necessariamente homogêneos. Atualmente, o grupo de Banco de Dados do IME-USP, participa de diversos projetos de pesquisa relacionados a análise de dados clínicos e moleculares. Cada um desses projetos possui muitos gigabytes de dados e, a cada mês, mais dados chegam. Era sabido que utilizar um DW para integrar esses dados seria a melhor solução, no entanto, devido a grande quantidade de dados inconsistentes e também, devido a grande freqüência com que os requisitos do sistema eram modificados, optou-se por construir um DW não incremental para cada projeto. Essa decisão de DW sem integração e não incremental, foi suficiente para sanar as necessidades do grupo de banco de dados temporariamente, no entanto, com o aumento do número de projetos, o volume cada vez maior de dados e as solicitações de análise cada vez mais frequentes, a construção de um único DW incremental tornou-se essencial.
O objetivo desse trabalho é desenvolver um DW incremental que integre todos os projetos de análise clínica e molecular em que o grupo de banco de dados do IME-USP está envolvido.

Palavras-Chave: Bancos de Dados, Data Warehouses, ETL.

2 Introdução

2.1 Contextualização

O Grupo de Banco de Dados (grupo de BD)do IME-USP participa de diversos projetos de armazenamento e análise de dados clínicos e moleculares. Normalmente, a participação do grupo nesses projetos é a de intermediador; em outras palavras, existem os pesquisadores que têm o conhecimento necessário para fazer inferências a partir de um conjunto de dados, existem os provedores que disponibilizam os dados brutos (com erros e/ou inconsistências) e o grupo de BD que recebe os dados desses últimos e os transformam para o formato esperado pelos pesquisadores. Essa transformação é aplicada aos dados provenientes de uma série de bancos de dados heterogêneos, basicamente de hemocentros e hospitais. Sua função é levar o dado a um estado confiável para que os pesquisadores e analistas possam fazer inferências mais precisas (Figura 1).

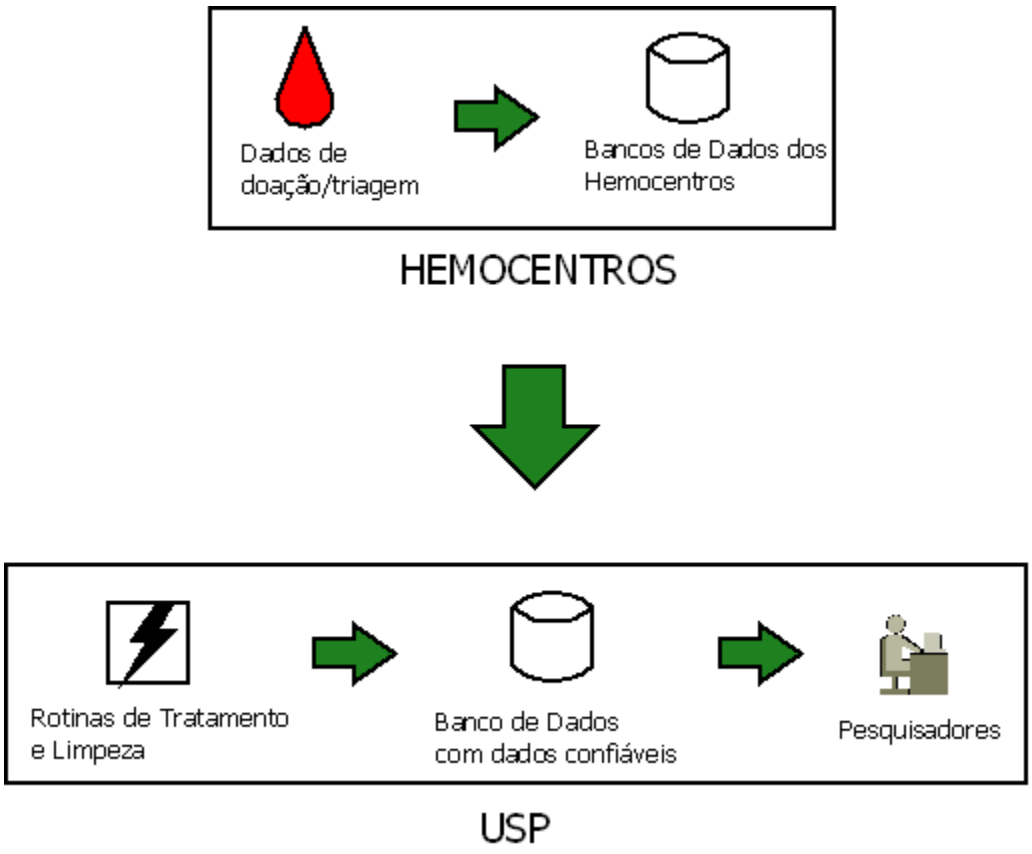


Figura 1: *Fluxo de dados padrão de um projeto de análise de dados clínicos/moleculares.*

2.2 Organização do Trabalho

- **Capítulo 3:** Descreve os principais projetos que serviram de motivação para este trabalho;
- **Capítulo 4:** Descrição sucinta dos conceitos chave utilizados no trabalho;
- **Capítulo 5:** Apresentação das etapas de desenvolvimento do sistema;
- **Capítulo 6:** Resultados da implementação;
- **Capítulo 7:** Conclusões.

3 Motivação

3.1 Projeto REDS-II

O projeto REDS-II (Retrovirus Epidemiology Donor Study 2) é uma iniciativa norte-americana da qual o grupo de BD participa desde 2006 e cujo tema principal é a segurança transfusional. Mensalmente, o hemocentro de São Paulo (Fundação Pró-Sangue), o hemocentro de Minas Gerais (Hemominas) e o hemocentro de Pernambuco (Hemope) enviam ao grupo de BD imagens de seus bancos transacionais que contêm os dados de todas as triagens (etapas do processo doação). A partir desses dados, deve-se aplicar uma série de transformações que resultarão em dados mais confiáveis para a geração de relatórios que serão analisados pelos pesquisadores, analistas e estatísticos envolvidos no projeto.

3.2 Solução para o projeto REDS-II

Ao analisar os requisitos do projeto REDS e as características dos dados dos hemocentros, logo percebeu-se que a melhor solução seria a utilização de um grande DW no intuito de integrar as fontes de dados. No entanto, havia uma grande necessidade por parte dos grupos de pesquisa pela geração rápida de resultados e, como a modelagem de um DW e de rotinas ETL é algo que exige muito tempo e grande intimidade com os dados da fonte, optou-se por uma solução que pudesse ser construída em pouco tempo: um conjunto de pequenos DWs não incrementais, ou seja, que não podem ser atualizados, um para cada hemocentro.

Essa solução permitiu que os relatórios para o projeto REDS-II fossem gerados em tempo. No entanto, trouxe diversas limitações, como a dificuldade de se analizar os dados dos três hemocentros juntos e a dificuldade de geração de dados históricos, uma vez que o DW não foi criado para ser atualizado.

As rotinas ETL de cada um dos DWs também continham algumas limitações importantes. Neste caso, isso aconteceu principalmente devido a escolha da ferramenta de desenvolvimento, o Microsoft SQL Server 2000, que apesar de ser uma das melhores opções na época, não oferecia a flexibilidade

que seria necessária para dar manutenção em um sistema tão complexo.

Como a parceria com os hemocentros estava apenas começando, a alteração, inclusão e remoção de atributos no conjunto de dados que chegava ao grupo de BD era muito frequente. Entretanto, o pacote SQL Server 2000 tem uma característica estática, ou seja, mudar configurações e acrescentar ou retirar transformações em uma rotina já existente são ações burocráticas e difíceis de ser executadas, logo, era comum que um programador demorasse horas, até dias para conseguir contornar um problema aparentemente simples.

A figura 2, na página seguinte, demonstra esquematicamente o fluxo dos dados desde o hemocentro até a aplicação dos resultados dessas pesquisas à sociedade.

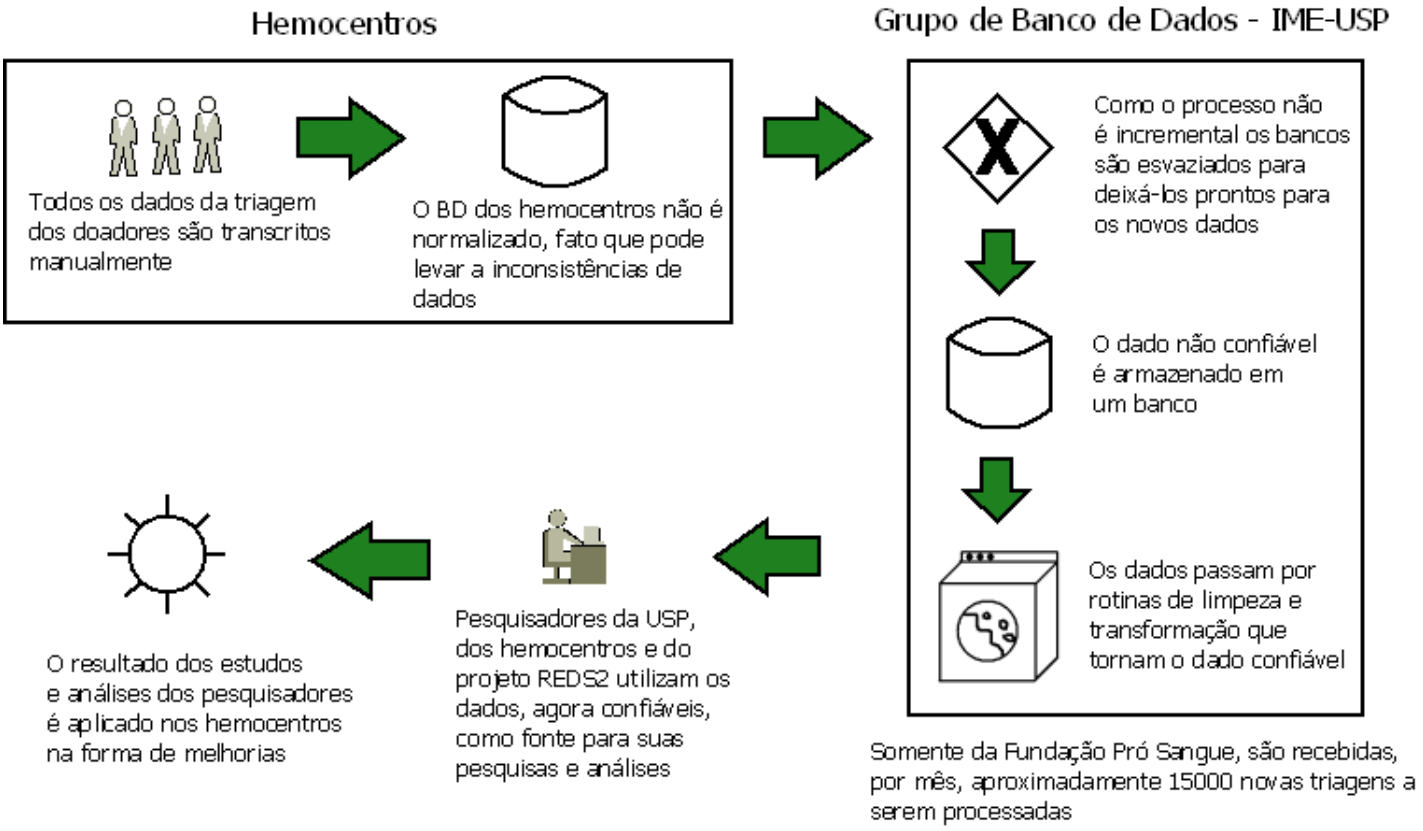


Figura 2: *Fluxo de dados padrão de um projeto de análise de dados clínicos/moleculares.*

3.3 Outros projetos que utilizam dados dos DWs

Não é apenas o projeto REDS-II que faz uso dos dados dos DWs dos hemocentros. Para exemplificar as limitações da solução que é utilizada atualmente e a motivação deste trabalho, serão citados aqui dois projetos que têm como finalidade a análise de dados históricos. O primeiro é uma

parceria do grupo de BD com a Fundação Pró-Sangue e que estuda a história de um certo indivíduo após sua doação ou transfusão (recepção) de sangue. Para isso, é feito um "*matching*" de outros bancos com dados clínicos e de óbitos com o DW da Fundação Pró-Sangue. O problema é que, como existem dados da Fundação desde 1996 e o DW não é incremental, a única forma de garantir que um indivíduo não está presente na fonte de dados é gerando, mês a mês desde 1996 até hoje, todas as cargas do DW.

Outro bom exemplo é o projeto, também em parceria com a Fundação Pró-Sangue, que estuda características epidemiológicas dos doadores de sangue que após múltiplas doações vieram a desenvolver anemia. Mais uma vez, como saber se um determinado indivíduo teve anemia se o DW não mantém dados históricos? A solução, como no projeto anterior foi gerar mês a mês, desde 1996, cargas do DW e armazenar essa informação em um banco de dados separado, para que a informação não fosse apagada no próximo processo de carga.

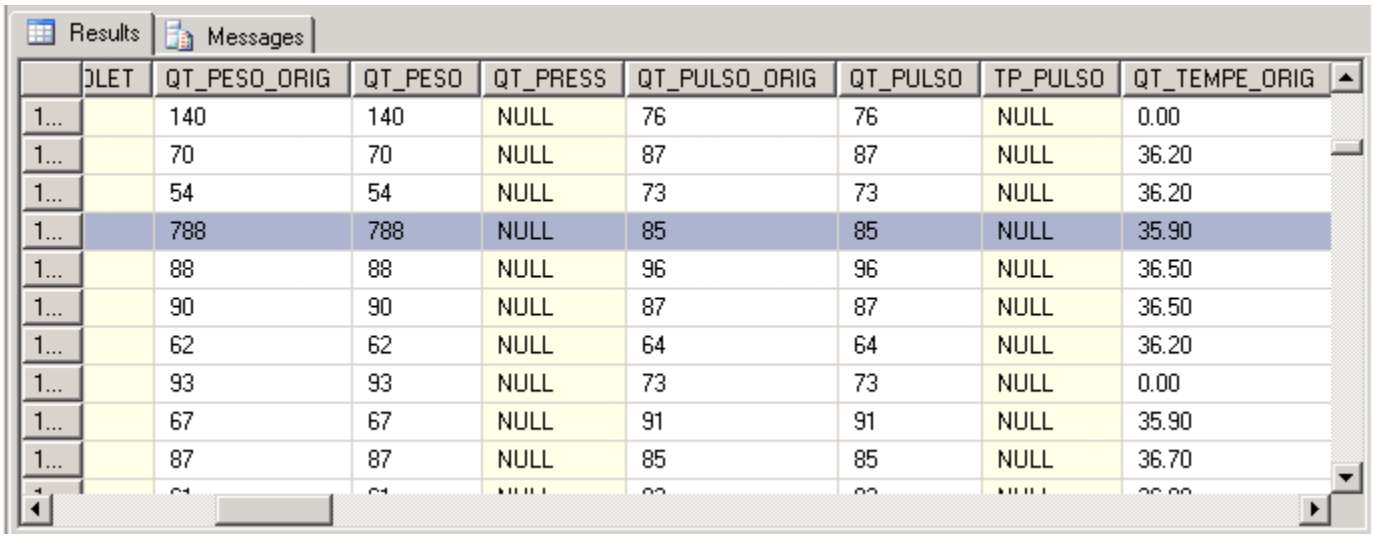
3.4 Projeto em parceria com o Ministério Da Saúde

Recentemente, o grupo de BD iniciou um novo projeto em parceria com o Ministério Da Saúde que visa a integração dos dados dos hemocentros brasileiros. Inicialmente participarão quatro novos hemocentros, são eles: Hemocentro do Rio de Janeiro, Hemocentro de Santa Catarina, Hemocentro do Amazonas e Hemocentro de Brasília. A participação do grupo de BD nesse projeto impulsionou a implementação de um DW incremental e de rotinas ETL de fácil manutenção, uma vez que já em janeiro de 2010 começarão a ser introduzidos os novos dados desses hemocentros e seria muito difícil utilizar a solução do projeto REDS-II para manter esse volume tão heterogêneo de dados.

3.5 Formato dos dados enviados pelos hemocentros

Os hemocentros enviam ao grupo de BD um conjunto de arquivos de texto que são, na verdade, a imagem de seus bancos de dados transacionais. A necessidade de rotinas ETL sobre esses dados se deve, além da heterogeneidade das bases, a erros e inconsistências devido, principalmente, a inserção manual de dados feita por funcionários. Além disso, esses bancos, em sua maioria, não são normalizados [3][1]. Para exemplificar o que cada um desses dois problemas citados anteriormente resultam, serão utilizados trechos reais dos arquivos enviados pela Fundação Pró-Sangue de São Paulo (foram editados nomes e chaves devido a confidenciabilidade de dados médicos, nos termos da lei).

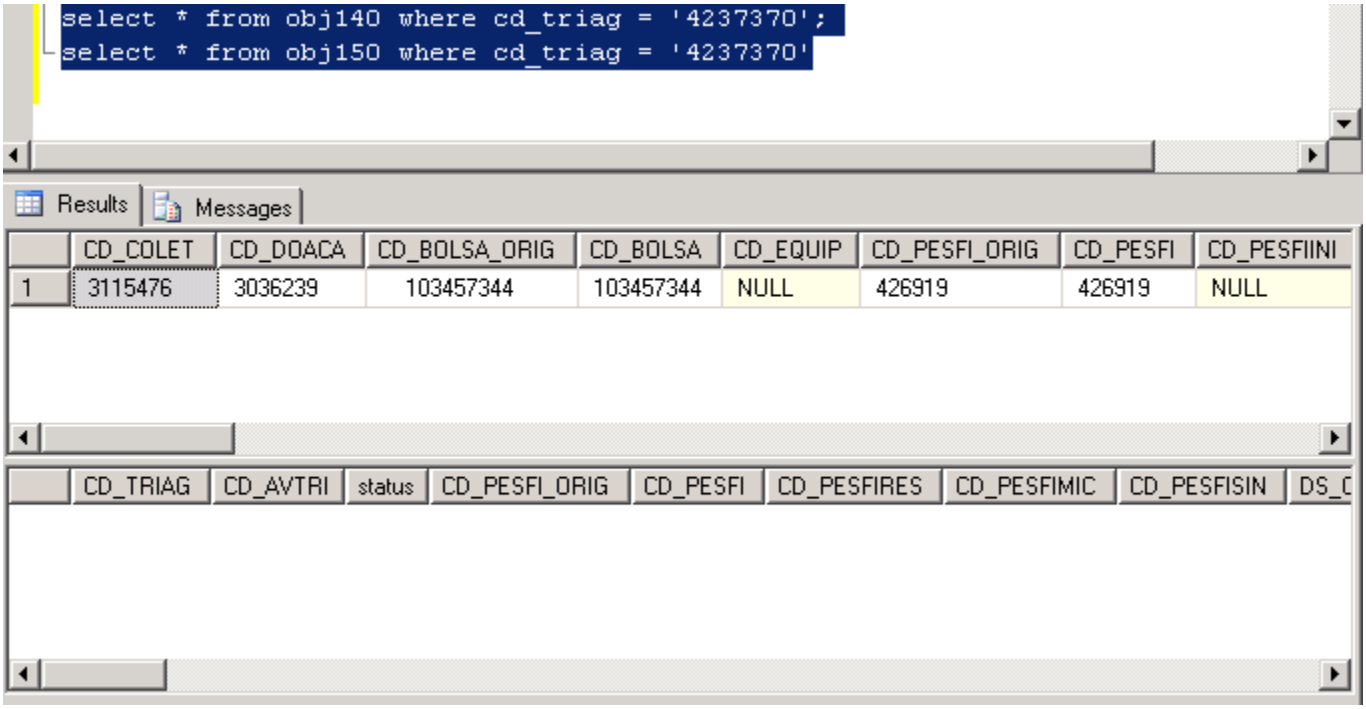
As figuras 3 e 4 mostram dois erros bastante comuns encontrados nos dados fonte enviado pelos hemocentro: Erros de digitação e erros de inconsistência.



Results | Messages

	)LET	QT_PESO_ORIG	QT_PESO	QT_PRESS	QT_PULSO_ORIG	QT_PULSO	TP_PULSO	QT_TEMPE_ORIG
1...		140	140	NULL	76	76	NULL	0.00
1...		70	70	NULL	87	87	NULL	36.20
1...		54	54	NULL	73	73	NULL	36.20
1...		788	788	NULL	85	85	NULL	35.90
1...		88	88	NULL	96	96	NULL	36.50
1...		90	90	NULL	87	87	NULL	36.50
1...		62	62	NULL	64	64	NULL	36.20
1...		93	93	NULL	73	73	NULL	0.00
1...		67	67	NULL	91	91	NULL	35.90
1...		87	87	NULL	85	85	NULL	36.70

Figura 3: *Erro de digitação na coluna qtpeso, que mostra um candidato com 788 Kg.*



```
select * from obj140 where cd_triag = '4237370';
select * from obj150 where cd_triag = '4237370'
```

Results | Messages

	CD_COLET	CD_DOACA	CD_BOLSA_ORIG	CD_BOLSA	CD_EQUIP	CD_PESFI_ORIG	CD_PESFI	CD_PESFIINI
1	3115476	3036239	103457344	103457344	NULL	426919	426919	NULL

	CD_TRIAG	CD_AVTRI	status	CD_PESFI_ORIG	CD_PESFI	CD_PESFIRES	CD_PESFIMIC	CD_PESFISIN	DS_C

Figura 4: *Exemplo de inconsistência: um indivíduo está presente na tabela que contém os doadores, mas não está na tabela das triagens. Isso é impossível.*

3.6 Objetivo

O objetivo desse trabalho é desenvolver um tipo particular de banco de dados chamado Data Warehouse que seja capaz de integrar todas as fontes de dados clínicos e moleculares, além das rotinas necessárias para carregar os dados das fontes e transformá-los em um formato pré-estabelecido. Como o tempo para o projeto e implementação é de pouco mais de cinco meses, decidiu-se que essas rotinas de transformação seriam implementadas apenas para o Hemocentro de São Paulo/Pró-Sangue, que é o maior hemocentro do Brasil e que envia mensalmente em torno de 200MB de dados para os bancos do grupo de BD.

A figura 5 descreve esquematicamente os objetivos deste trabalho.

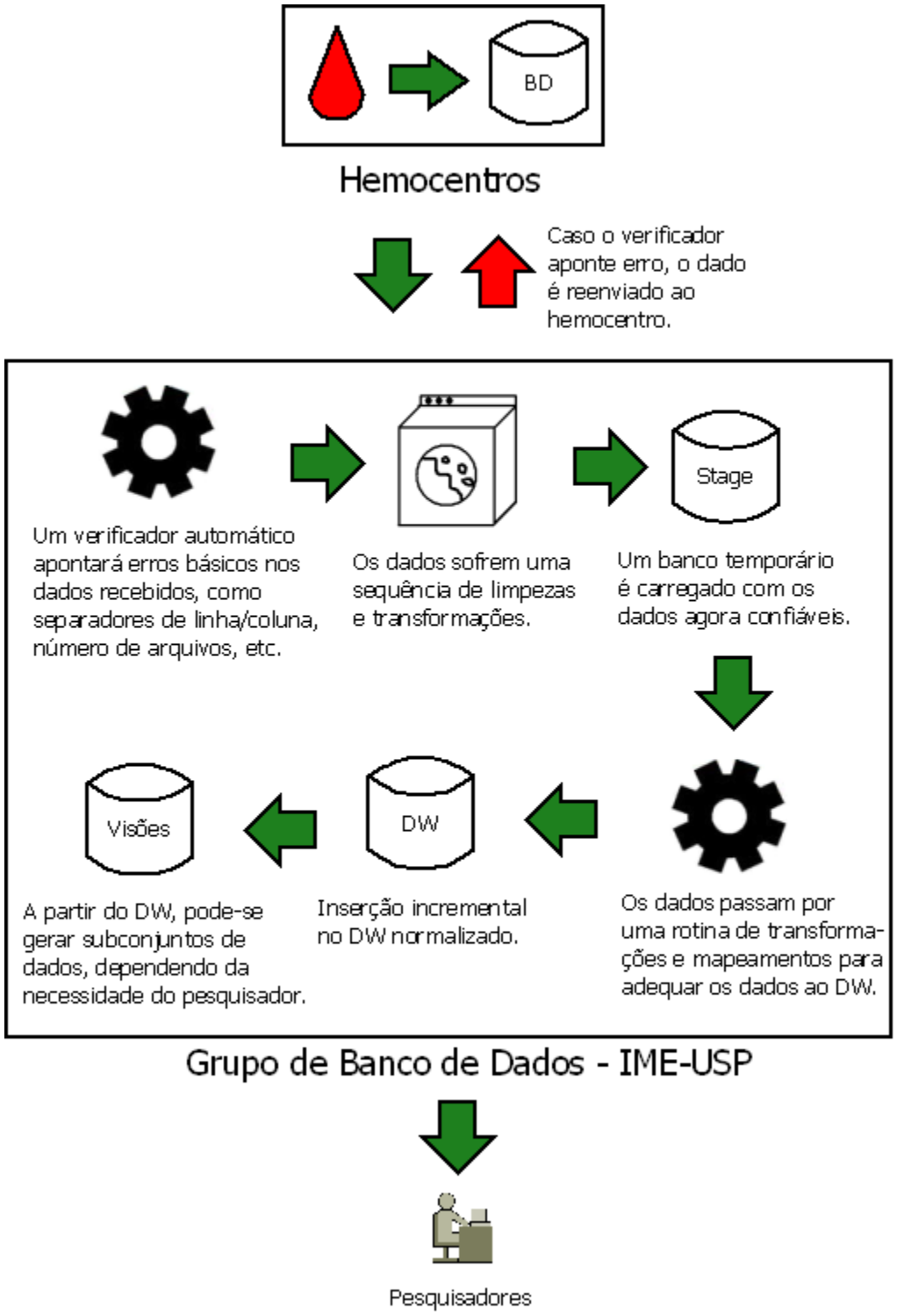


Figura 5: *Objetivos.*

4 Conceitos

4.1 Modelo Conceitual

Em [1] modelo conceitual é definido como "uma descrição concisa dos requisitos dos usuários que inclui descrições detalhadas dos tipos de entidade, relacionamentos e restrições e que são expressas usando os conceitos de um modelo de dados de auto nível".

Quando um projeto de banco de dados é iniciado, o primeiro passo a ser feito é fazer um levantamento detalhado dos requisitos dos usuários que irão utilizá-lo. Para descrever o modelo resultante desses estudos sem ter que se preocupar com detalhes de implementação (como tipos de dados, estruturas de dados, rotinas SQL) deve-se utilizar uma abstração minimalista mas que descreva o comportamento do banco, em outras palavras, um modelo conceitual.

No caso desse trabalho, foi usado como ferramenta para descrever o modelo conceitual, o Modelo Entidade Relacionamento Estendido (MER-X) [3][1].

4.2 Data Warehouses

Organizações ou empresas que têm grande produção de dados, em algum momento, têm que decidir como tratar esse excesso de informação para que não haja perda de desempenho na execução de consultas aos seus bancos transacionais. Muitas vezes, existe também a necessidade de se tomar decisões de negócios e, para isso, é necessário que haja uma boa quantidade de dados históricos para que seja possível levantar estatísticas e encontrar tendências na informação. Novamente, a utilização dos bancos de dados transacionais para tais fins levaria a perdas significativas de desempenho dependendo do tamanho do conjunto de dados e no nível de requisição do banco.

Os Data Warehouses apareceram em meados dos anos 80 para preencher o vão entre a necessidade de se fazer inferências na informação e a necessidade de diminuir a carga nos bancos transacionais dos dados que são pouco acessados sem que se perca a "história" deles. É importante salientar que a informação pode provir de diversos bancos transacionais totalmente diferentes entre si, logo, percebe-se outra característica importante de Data Warehouses, que é a de integrar bases heterogêneas de dados [13][14][1].

Em termos mais formais, um Data Warehouse é um sistema normalmente composto por uma grande estrutura de tabelas que possibilita a integração de informação proveniente de uma ou mais fontes de dados heterôgeneas (possivelmente bancos de dados transacionais) para posterior análise.

Durante o texto será empregado o termo Data Warehouse ou DW e não sua tradução Armazém de Dados ou Depósito de Dados porque o primeiro já é consagrado em artigos e livros escritos em língua portuguesa.

Para a construção de DWs são utilizadas duas arquiteturas principais; a dimensional e a normalizada.

4.2.1 Arquitetura Dimensional

O modelo dimensional representa os indicadores importantes para uma área de negócios, que são chamados de fatos ou métricas e os parâmetros, chamados dimensões, através dos quais estas métricas são vistas [3]. Este modelo é a base da arquitetura dimensional, que é utilizada para o desenvolvimento de DWs. Por ter como característica principal a desnormalização, essa arquitetura pode ser implementada rapidamente e, pela simplicidade de sua estrutura, o desempenho das consultas feitas aos DWs que usam essa estrutura constuma ser melhor do que com as outras arquiteturas [5][13]. Por outro lado, a desnormalização também implica em dificuldade de manutenção, inconsistências e dificuldade de integração de fontes diferentes, especialmente quando o volume de dados é muito grande [13][14].

4.2.2 Arquitetura Normalizada

Na arquitetura normalizada, diferentemente da arquitetura dimensional, os bancos de dados que constituem o DW estão normalizados (terceira forma normal ou superior) e consequentemente menos vulneráveis a inconsistências. Apesar de ser mais difícil de se projetar e de exigir rotinas ETL mais complexas, a utilização dessa arquitetura é mais flexível do que a dimensional, uma vez que, a partir do DW normalizado é possível gerar diversas visões dos dados, inclusive no modelo dimensional.

4.3 ETL

ETL é proveniente do inglês *Extract, Transform and Load*, ou seja, Extração, Transformação e Armazenamento. Em linhas gerais, esse é o principal método para se carregar dados para um Data Warehouse: primeiro a informação deve ser recuperada da fonte de dados (Extração); então, uma série de regras de tratamento, limpeza e mapeamento são executadas sobre os dados (Transformação) para só após serem inseridos no DW (Armazenamento). Esse processo é chamado de extrair, transformar e armazenar os dados de processo de carga.

5 Metodologia

5.1 Avaliação das ferramentas disponiveis

Quando o projeto teve início, havia a necessidade de se escolher um gerenciador de bancos de dados e um ambiente de desenvolvimento de rotinas ETL. Após algumas semanas pesquisando as ferramentas disponíveis no mercado, foram selecionadas duas que mais se aproximavam das necessidades deste trabalho: o pacote Pentaho (código aberto)[12] e o pacote Microsoft SQL Server 2008 [7].

5.1.1 Pacote Pentaho

Em 2008 o então aluno de doutorado Márcio K. Oikawa, teve como condição para apresentar um de seus trabalhos [10] em um congresso internacional, a migração de todo um sistema de Data Warehouse para alguma plataforma de código aberto. Basicamente, foi preciso migrar uma base de dados e suas ferramentas ETL do pacote SQL Server 2000 para alguma ferramenta de código aberto. Para migrar a base de dados foi simples, afinal, existem gerenciadores de banco de dados de código aberto ou gratuitos já consagrados, pela internet. Assim, foi escolhido o Postgres SQL [2].

No caso do ambiente desenvolvedor de rotinas de ETL, houve bastante dificuldade, porque a maioria das soluções encontradas não era estável, ou não tinha uma comunidade ativa (suporte) ou não tinha uma interface amigável. Depois de bastante procura, foi encontrada uma ferramenta que parecia ser confiável, que tinha uma comunidade ativa de desenvolvimento e que era de código aberto. Esse ambiente de desenvolvimento chama-se Pentaho Data Integration e é uma ferramenta visual fortemente baseada nas soluções do SQL Server 2005 e 2008. Todas as rotinas ETL utilizadas para carregar o DW a partir dos bancos de dados relacionais fonte foram feitas com sucesso utilizando essa ferramenta.

Para finalizar, era necessário encontrar um visualizador de consultas OLAP que substituisse os relatórios dinâmicos que eram gerados pelo SQL Server e visualizados no Microsoft Excel. A melhor solução encontrada foi um programa chamado JPivot que funcionava em conjunto com o Pentaho Mondrian [11], e que, além de gerar a exibição das consultas OLAP, fazia isso em um ambiente web.

O trabalho foi aceito na conferência e foi decidido estender a utilização das ferramentas para serem testadas com outros projetos do grupo BD. Depois da apresentação para usuários que não eram programadores, percebeu-se que a solução Open Source não era capaz de agradar aos pesquisadores devido à interface complicada e não amigável do Jpivot [4].

As figuras 6 e 7 são *screenshots* da ferramenta de desenvolvimento de rotinas ETL do pacote Pentaho e do JPivot, respectivamente.

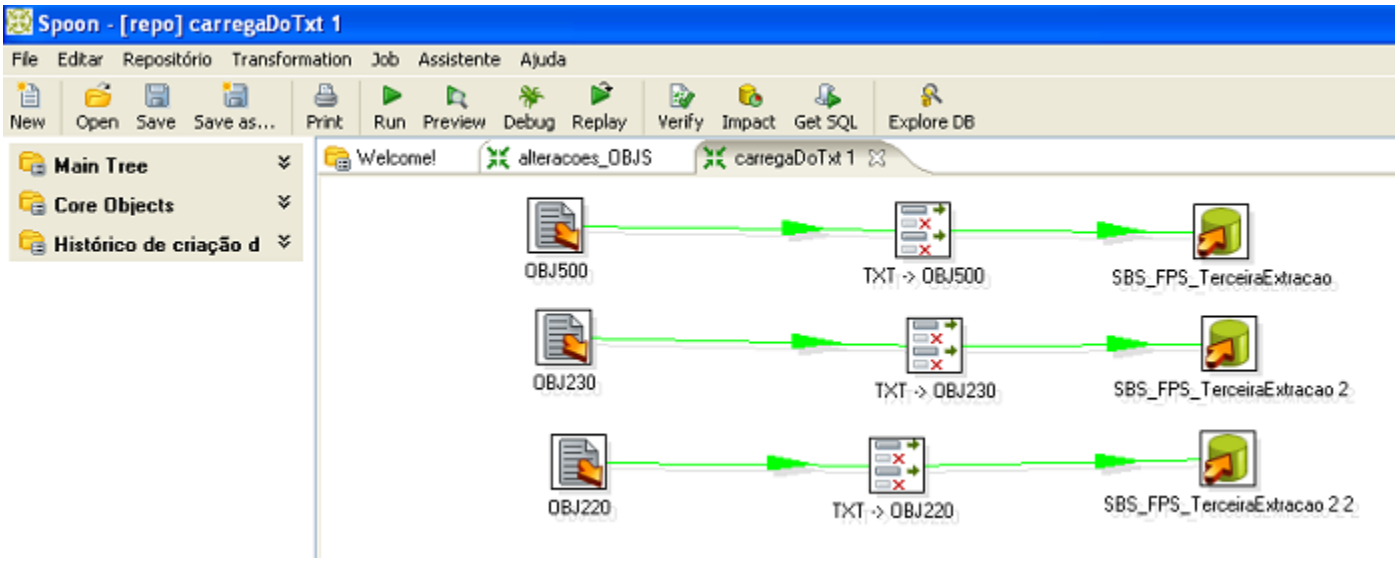
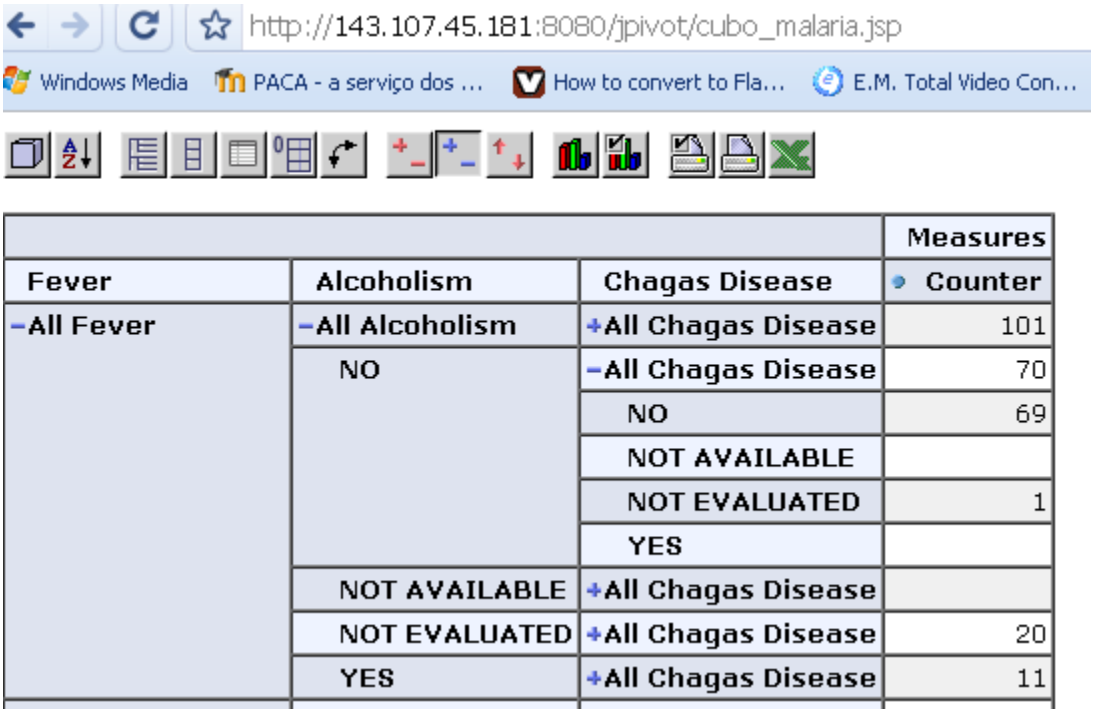


Figura 6: *Ambiente de desenvolvimento do Pentaho Data Integration.*



			Measures
Fever	Alcoholism	Chagas Disease	Counter
-All Fever	-All Alcoholism	+All Chagas Disease	101
	NO	-All Chagas Disease	70
		NO	69
		NOT AVAILABLE	
		NOT EVALUATED	1
		YES	
	NOT AVAILABLE	+All Chagas Disease	
	NOT EVALUATED	+All Chagas Disease	20
	YES	+All Chagas Disease	11

Figura 7: *JPivot sendo executado em um navegador.*

5.1.2 Pacote Microsoft SQL Server 2008

O pacote SQL Server 2008 é a última versão do SQL Server, que chamou a atenção do grupo BD pela série de melhorias que eram discutidas em blogs e sites especializados. As melhorias que mais chamou a atenção do grupo foram as relacionadas à manutenção das rotinas:

- Suporte a utilização de variáveis e funções para configurações das rotinas;
- Integração com o C# [8];
- Melhorias na interface;
- Inserção de "componentes" novos, como condicionais, ferramentas de *"look-up"*, etc.

Além disso, o SQL Server é capaz de gerar relatórios dinâmicos que são visualizados pelo Microsoft Excel que é amplamente utilizado por pesquisadores.

Durante aproximadamente duas semanas, foram feitos uma série de testes para verificar os limites dessa ferramenta até então desconhecida. Assim, chegou-se a conclusão de que devido às melhorias citadas acima e também por seu sistema de ajuda e suporte online [6] seria melhor utilizá-la em vez da ferramenta de código aberto.

A figura 8 e 9 abaixo são *screenshots* do ambiente de desenvolvimento de rotinas ETL do Microsoft SQL Server 2008 e da visualização dos relatórios dinâmicos no Excel.

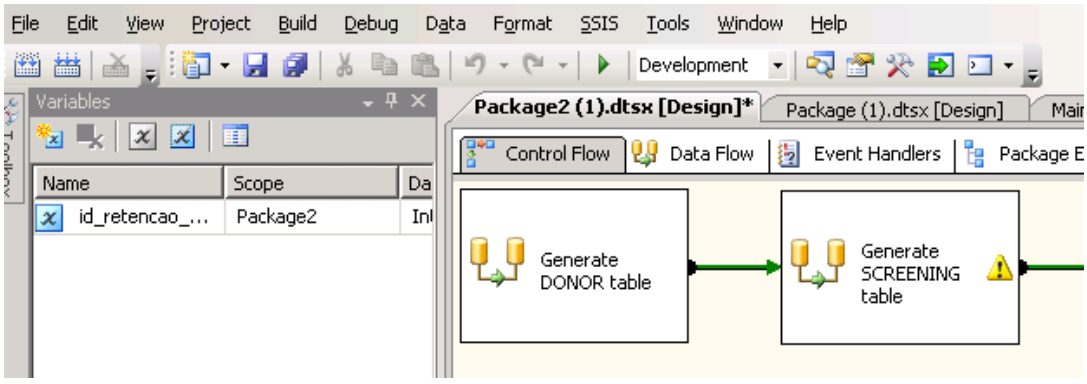


Figura 8: *Ambiente de desenvolvimento do Microsoft SQL Server2008.*

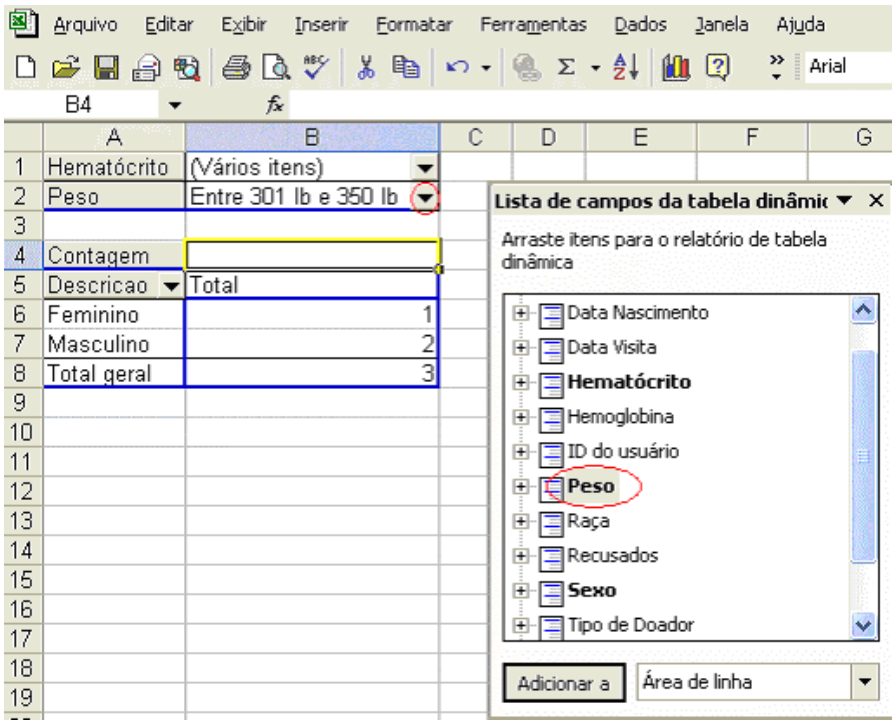


Figura 9: *Relatório dinâmico sendo visualizado no Microsoft Excel.*

5.2 Levantamento dos requisitos do DW

Para se modelar um DW deve-se, antes de tudo, entender qual será sua finalidade. Deve-se levantar cuidadosamente seus requisitos, ou seja, saber qual o conjunto de dados necessários por cada um dos projetos e fazer projeções de dados que podem ser necessários no futuro [1][3]. Neste projeto, foi utilizado como base principal os dados dos relatórios enviados ao projeto REDS-II, que acabam englobando praticamente todos os conjuntos de dados dos outros projetos.

5.3 Projeto do modelo conceitual

Uma vez estabelecidos a finalidade e o conjunto de dados disponibilizado pelo DW é preciso entender, quais dados da fonte devem ser utilizados para gerar o DW. Por exemplo, existem atributos que são gerados a partir de dois ou três campos diferentes; ou então, existem atributos que podem ter um significado diferente em duas fontes de dados distintas. Além disso, existem dados que tem prioridade sobre outros, como no caso dos resultados de exame para o projeto REDS-II, onde saber o resultado de um exame de HIV é muito mais importante do que saber se o indivíduo estava anêmico ou saber o nome do enfermeiro que o atendeu. Resumindo, é necessário conhecer muito bem os dados provenientes das fontes de dados pois só assim será possível garantir que os dados do DW foram extraídos do lugar correto.

5.4 Procedência de dados

Determinado o conjunto de dados proveniente das fontes de dados, é possível, então, planejar, para cada atributo, quais etapas de transformação serão necessárias para levar os dados das fontes ao estado projetado na fase de levantamento de requisitos. Mais uma vez, o grande conhecimento dos dados é essencial, pois apenas dessa forma é possível descrever a procedência dos dados, ou seja, somente assim pode-se saber de qual tabela um determinado atributo provém, a que época pertence e quais problemas podem afetar este campo durante esse período [1][13].

Um exemplo que ilustra como é importante a procedência de dados, foi o caso de alguns indivíduos presentes nos arquivos da Fundação Pró-Sangue que tinham os valores dos atributos de níveis de hematócrito e hemoglobina inválidos. Era sabido que durante o período que ocorreu esse problema, apenas os níveis de hematócrito haviam sido medidos. Entretanto percebeu-se que para alguns indivíduos haviam registros como, por exemplo, zero para níveis de hematócrito e níveis altíssimos de hemoglobina (por volta de 40g/dl, quando a consideração de normalidade é em torno de 15.0 g/dl). Com isso, percebeu-se que havia acontecido uma troca no momento em que o funcionário passou os dados no computador. Portanto, bastava uma rotina para inversão dos valores de hematócrito e hemoglobina dentro dessas circunstâncias.

5.5 Implementação do sistema do DW

Pode-se chamar de sistema do DW o conjunto de banco de dados que auxiliam a atualização do DW. Para este trabalho, decidiu-se que esse sistema seria composto pelos seguintes bancos de dados:

- Stage: O Stage é um conglomerado de bancos de dados e serve de passo intermediário entre os dados brutos, diretos da fonte e o DW normalizado. Dentro dele, temos dois bancos temporários, a imagem do banco transacional de um hemocentro, neste caso, do hemocentro de São Paulo e uma cópia do DW normalizado. Com exceção à rotina ETL que carrega os dados incrementalmente no DW, todas as rotinas são executadas sobre dados do Stage; daí sua existência. Isso é feito para que em caso de problemas, dados inválidos não sejam inseridos no DW.

- NDS (Normalized Data Store): É o banco de dados que contém o DW normalizado, propriamente dito.

  Uma vez determinados os requisitos do DW, e o modelo conceitual, pode-se partir para sua implementação. A partir do modelo, é possível projetar as tabelas e seus atributos e então, utilizando as ferramentas do pacote SQL Server 2008, implementá-los.

5.6 Arquivos de metadados

Antes da explicação de como foram implementados o verificador de dados da fonte e as rotinas ETL, é necessário explicar sobre os arquivos de metadados, que foram a solução encontrada para sanar o problema da heterogeneidade dos dados de um mesmo hemocentro. É muito comum que, com o passar do tempo, os procedimentos adotados por algum hemocentro mudem ou evoluam.

Foi criada uma linguagem simples de metadados que descrevem características de conjuntos de dados da fonte de acordo com um certo período. Por exemplo, todos os arquivos enviados ao grupo de BD em 2007 têm as mesmas características, então, segundo essa linguagem teríamos o seguinte:

Abaixo, há um trecho do arquivo principal de metadados, que descreve, a partir de um mês e um ano, a localização no sistema de arquivos dos metadados específicos para esse período.

```
from month=1 year=2007
to month=12 year=2007
use=/home/path/metadata2007.txt
```

O arquivo metadata2007.txt contém uma série de informações a respeito dos dados da fonte, como número de arquivos esperados, número de colunas de cada arquivo, separadores de coluna, etc.. Por exemplo:

center= Fundação Pro Sangue
initial_date= 01/2007
final_date= 12/2007
number_of_flat_files= 15
log_folder=C:/nds_bloodcenters/bloodcenters/fundacao_pro_sangue/log

flat_file_name= C:/nds_bloodcenters/bloodcenters/fundacao_pro_sangue/flat_files/OBJ010.TXT
number_of_columns= 24
column_separator= |
row_separator= \r\n
text_qualifier= ”

flat_file_name= C:/nds_bloodcenters/bloodcenters/fundacao_pro_sangue/flat_files/OBJ150.TXT
number_of_columns= 67
column_separator= |
row_separator= \r\n text_qualifier= ”

etc..

5.7 Implementação do Verificador de Dados da Fonte

Um dos objetivos desse trabalho é criar rotinas ETL para os dados provenientes do hemocentro de São Paulo/Pró-Sangue. Como já foi explicado anteriormente, esses dados chegam ao grupo de BD na forma de arquivos de texto que são um tipo de imagem do banco de dados transacional. É muito comum que alguns erros simples, mas difíceis de ser encontrados manualmente, interrompam o processo de carga e causem transtornos ao programador, que será obrigado a depurar o arquivo fonte para então resolver o problema ou, dependendo do caso, pedir o reenvio dos dados para o hemocentro.

Para solucionar esse tipo de problema, foi desenvolvido um verificador de arquivos em Java [9] que funciona da seguinte maneira:

- O mês, ano e nome do centro são passados como parâmetro no momento da execução do verificador.
- Baseado nos metadados específicos do período em questão, o programa faz a verificação dos arquivos da fonte.

- Caso encontre algum problema, o verificador tem um simplificado de Log que registra os erros encontrados para que o programador decida se vai corrigi-lo ou se deve solicitar uma nova remessa de dados ao hemocentro.

A idéia seria executar esse verificador em um sistema web onde os hemocentros poderiam fazer *upload* de seus dados, e assim, caso fosse encontrado algum erro, o próprio sistema enviaria um alerta para que os funcionários do hemocentro o corrigissem e então reenviassem a remessa. No entanto, devido à falta de tempo, foi decidido colocá-lo como o primeiro passo das rotinas ETL e terminar de implementar o sistema web futuramente.

5.8 Implementação das rotinas ETL

A ferramenta de desenvolvimento de rotinas ETL do pacote SQL Server 2008 é uma ferramenta visual e baseada inteiramente em Workflows [15]. Todas as ações acontecem dentro de componentes e a execução de cada uma delas depende do sucesso da execução anterior (Figura 10).

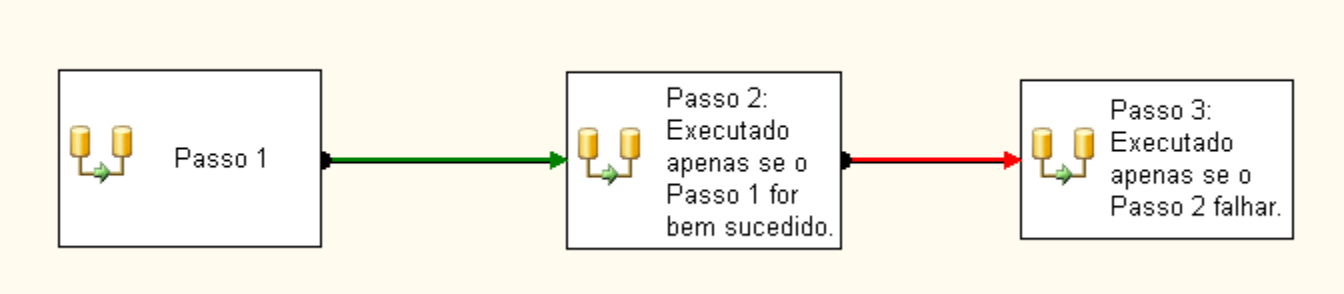


Figura 10: *Exemplo de um workflow no SQL Server 2008.*

A principal vantagem de utilizar o ambiente de desenvolvimento de rotinas ETL do pacote de algum sistema gerenciador de banco de dados é que os processos são otimizados. Isso é essencial, já que neste trabalho o ponto principal é a manipulação e tratamento de um grande volume de dados.

Após estudar e analisar a procedência dos dados foi possível conhecer quais atributos serão utilizados para gerar a carga do DW e por quais etapas de transformação cada um deles terá que ser submetido. Isso permitiu iniciar a implementação das rotinas ETL. Vale ressaltar que, como foi dito anteriormente, em razão do curto tempo para o desenvolvimento, decidiu-se gerar as rotinas somente para o hemocentro de São Paulo/Fundação Pró-Sangue, visto que este único hemocentro é responsável por 75% dos dados em posse do grupo de BD. Além disso, a Fundação Pró-Sangue apresenta uma heterogeneidade de dados razoável, uma vez que muitos procedimentos de triagem utilizados no passado foram sendo substituídos gradualmente por outros e assim, faz-se necessário que a rotina ETL esteja pronta para lidar com esse tipo de problema.

O primeiro passo do processo é executar uma rotina em C# que baseada no mês e ano definidos pelo programador e que foram armazenados em variáveis do próprio SQL Server, procura no arquivo principal de metadados pela localização do arquivo de metadados específico para o período. A partir daí, o programa verificador de dados da fonte é executado e caso nenhum problema seja encontrado, é permitido então começar a processar a carga.

Stage é o nome que foi dado ao banco temporário que serve de apoio para o envio de dados ao DW. Neste ponto, a rotina ETL é responsável por checar a validade de todas as chaves primárias dos arquivos da fonte e então armazená-los no Stage, que antes de todo esse processo, foi esvaziado. Tem-se então, uma imagem muito próxima da real (foram tiradas as tuplas com chaves inválidas) do banco transacional do hemocentro.

Até agora só foram verificadas as chaves de cada uma das tabelas. Como os bancos transacionais dos hemocentros não são normalizados, é preciso verificar as referências entre as tabelas para ver se existe alguma inconsistência, por exemplo, um indivíduo que nunca conseguiu efetuar uma doação de sangue e só deveria constar na tabela de recusas pode também estar na tabela de doadores. Feito isso temos certeza de que todas as referências estão corretas.

No Stage, existe uma versão temporária do DW normalizado que serve para facilitar a posterior atualização incremental. No momento, o que é necessário fazer é, dado o conjunto de dados da fonte, que agora estão corretamente referenciados entre si e com suas chaves primárias íntegras, deseja-se transformá-los e mapeá-los para o formato esperado pelo DW para então armazená-los no DW temporário. Um exemplo disso pode ser visto abaixo:

Por exemplo:

A figura 11 representa o processo de transformação do atributo gênero (gender). Primeiro é feito uma busca (ou lookup) com a tabela de gênero para verificar se o valor do atributo é válido. No caso do Hemocentro de São Paulo/Fundação Pró-sangue, serão aceitos somente os valores "*True*" e "*False*". Caso o atributo seja válido, é feito o mapeamento para o código do gênero esperado, no caso 1 ou 2; caso contrário, é verificado se o valor é desconhecido ou vazio, e então mapeado para o respectivo código, neste caso, 9 ou 8, respectivamente. Depois de executar mapeamentos desse tipo para todos os atributos, teremos o DW temporário com os dados prontos para serem usados para atualizar o DW funcional.

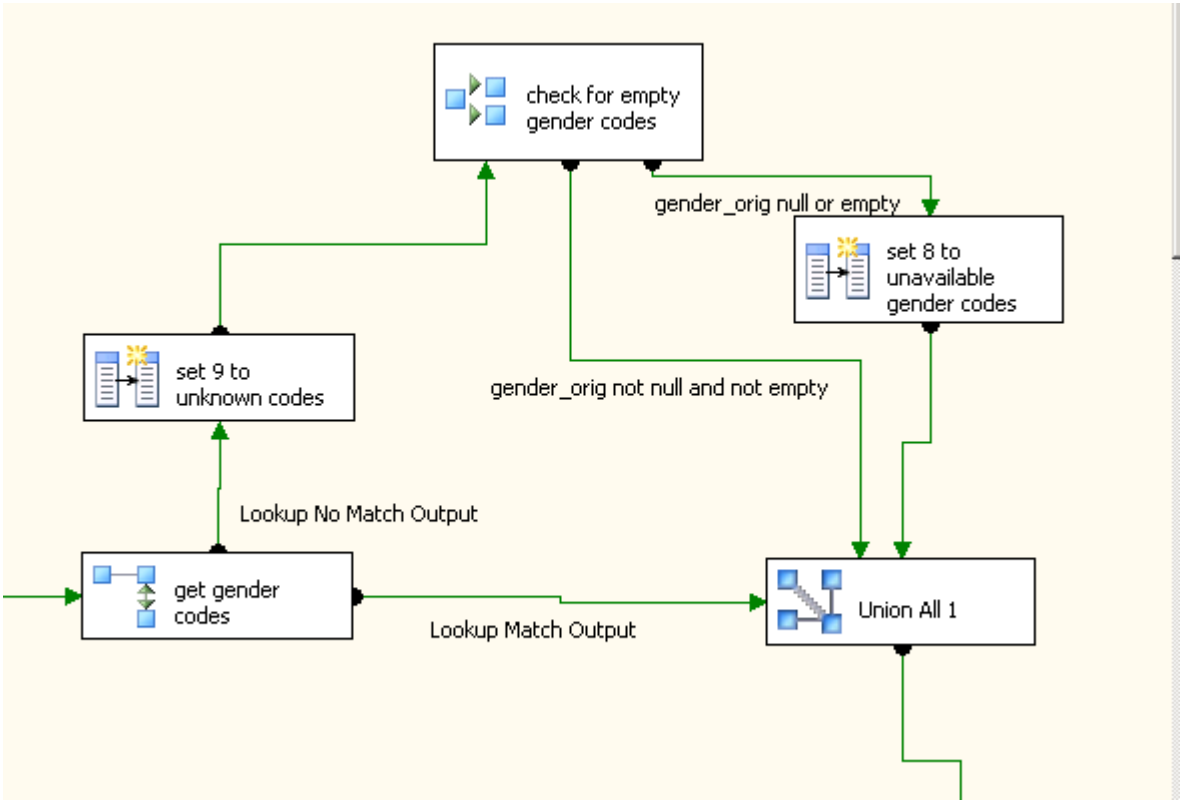


Figura 11: *Processo de transformação do atributo sexo (gender).*

A atualização funciona, basicamente, baseada na data em que ocorreu a triagem. Ao tentar inserir uma nova tupla, a rotina ETL verifica se já existe um dado antigo no DW; caso não haja, as tuplas são simplesmente inseridas, caso haja, compara-se as datas e, caso os dados presentes sejam mais antigos, eles são atualizados. O programador pode definir uma variável no ambiente de desenvolvimento que descreve se o DW deve ser atualizado ou então se permitirá apenas a inserção de novos dados. Isso se deve ao fato de que o hemocentro pode enviar dados antigos e não faz sentido sobreescrever os dados mais atualizados que estão no DW. Por exemplo, caso sejam disponibilizados dados de 1990, não faz sentido sobreescrever dados de 2009.

6 Resultados

6.1 Modelo conceitual

A figura 12 é o modelo conceitual resultante sem a representação dos atributos de cada entidade devido a limitações de espaço disponível. O modelo completo pode ser encontrado aqui.

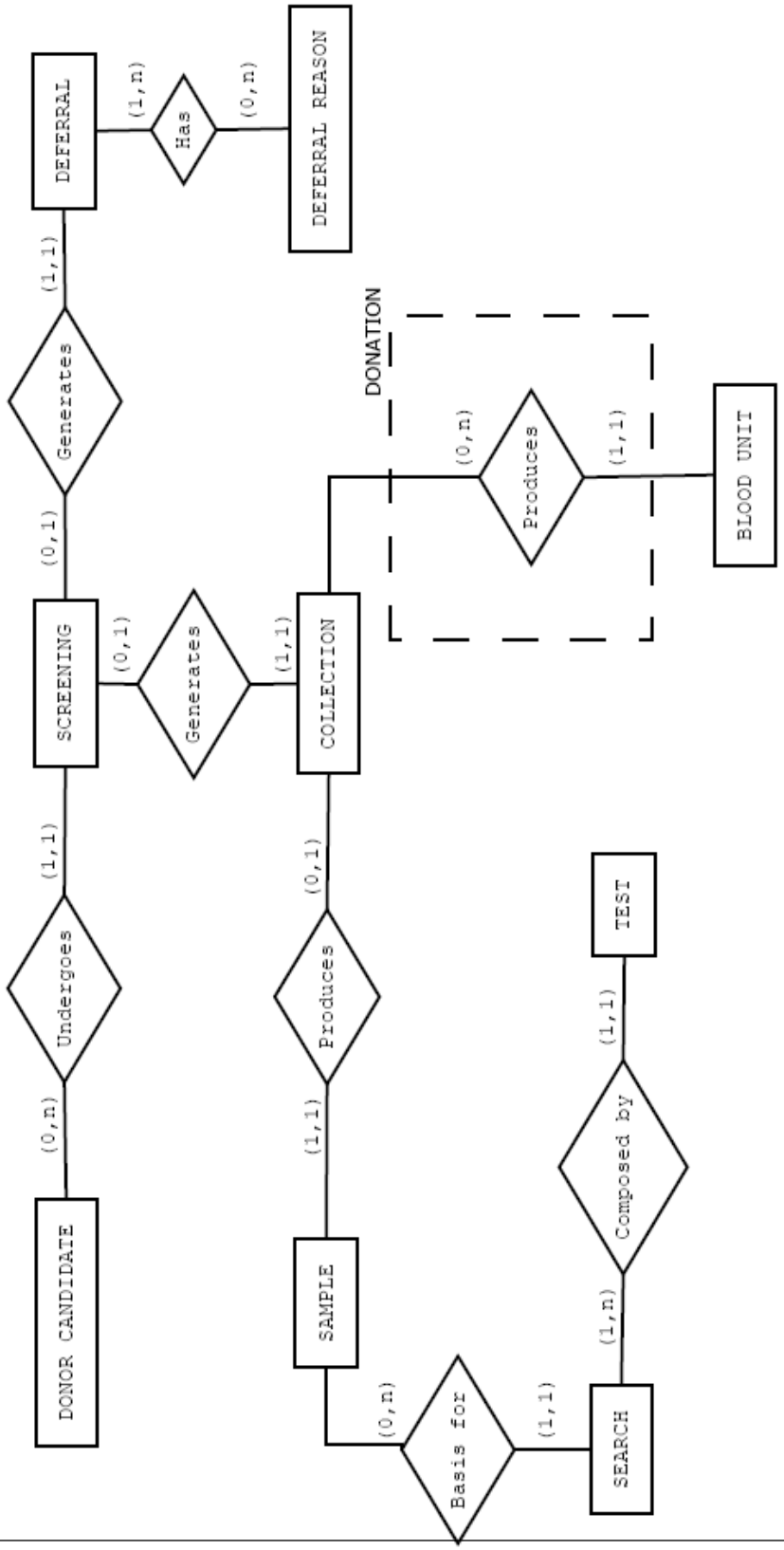


Figura 12: *Modelo Conceitual sem os atributos das entidades.*

6.2 Sistema de Data Warehouse Incremental

Abaixo (Figura 13) segue um modelo esquemático do que foi implementado, conforme descrito no cápitulo "Metodologias".

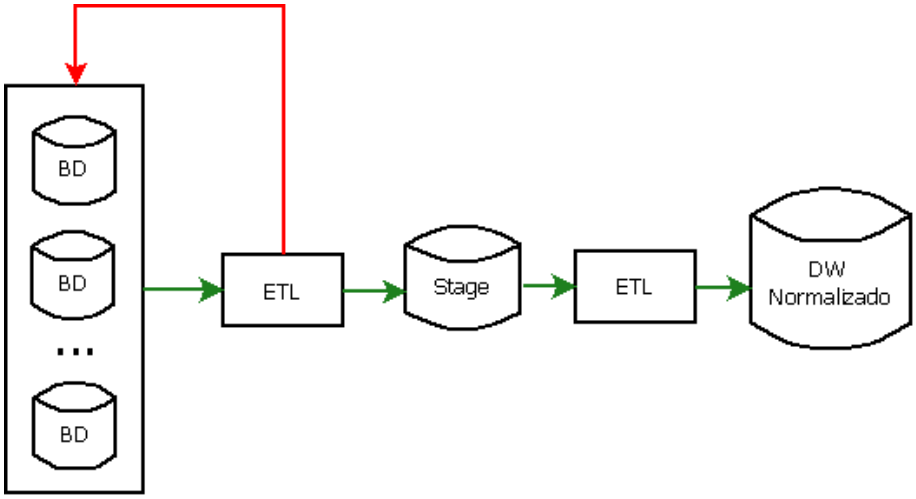


Figura 13: *Modelo Conceitual sem os atributos das entidades.*

6.3 Implementação das rotinas ETL

Como já foi explicado, devido a limitações de tempo foi decidido que seriam implementadas as rotinas ETL somente para o hemocentro de São Paulo/Fundação Pró Sangue. Com relação à implementação, houve a implementação de basicamente 4 rotinas, são elas:

- "*Main*" - (Principal) - que é responsável por fazer a chamada da execução das outras rotinas (Figura 14);
- "*Load Objs*" - (Carrega arquivos fonte) - que é responsável por carregar os dados da fonte e verificar as consistências de referência (Figura 15 e 16).Todas essas consistências foram descritas antes da implementação das rotinas;
- "*Populate Temporary NDS*" - (Carrega DW normalizado temporário) - que é responsável por transformar e mapear os dados para o DW normalizado temporário, que fica no Stage (Figura 17);
- "*Upsert data into NDS*" - (Atualiza ou insere dados no DW) - que é responsável por inserir os dados de forma incremental no DW (Figura 18);

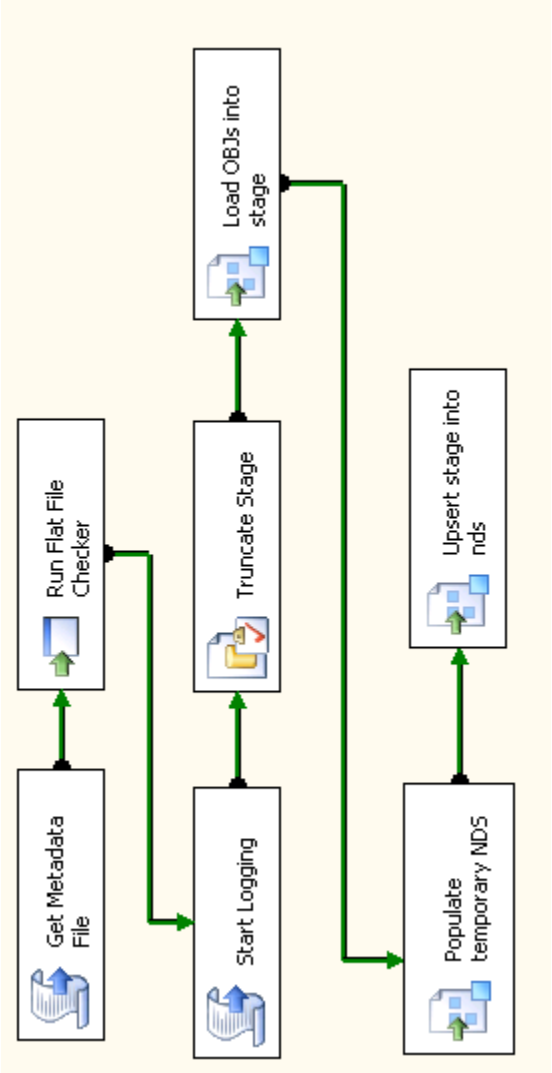


Figura 14: *Rotina "Main".*

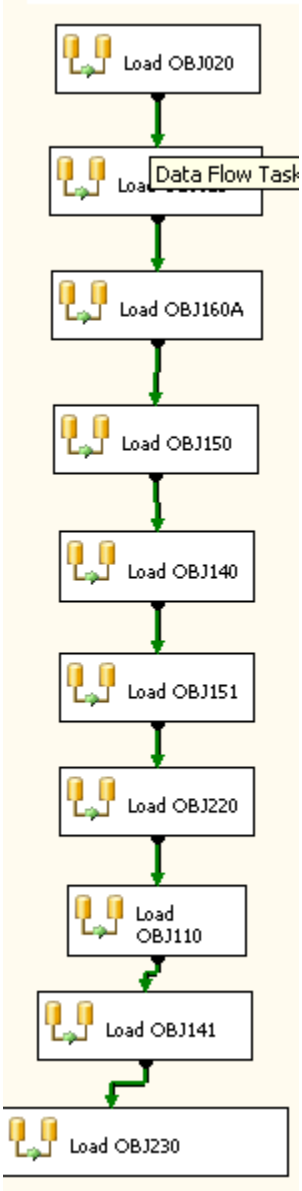


Figura 15: *Rotina "Load Objs"*.

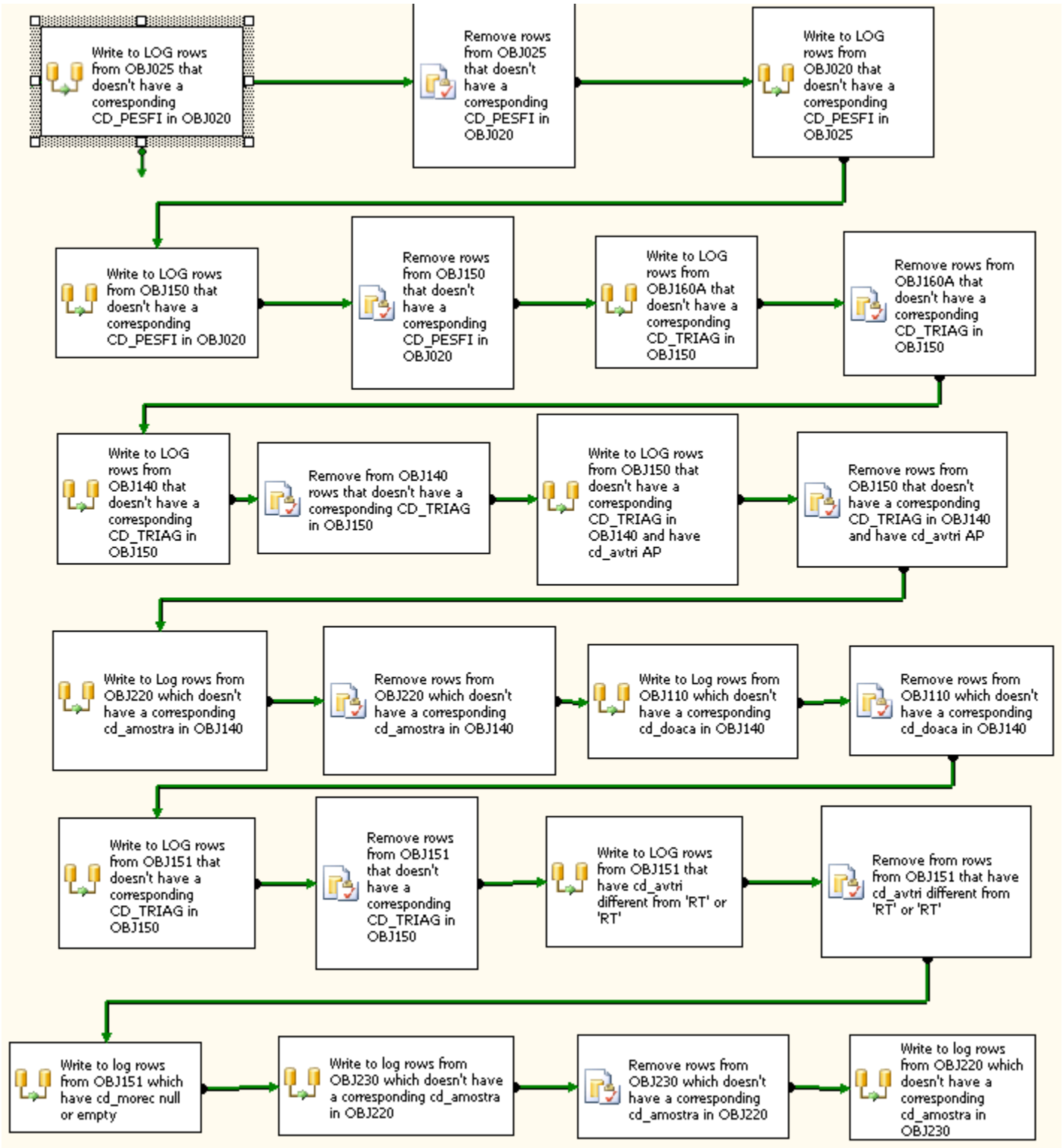


Figura 16: *Rotina "Load Objs".*

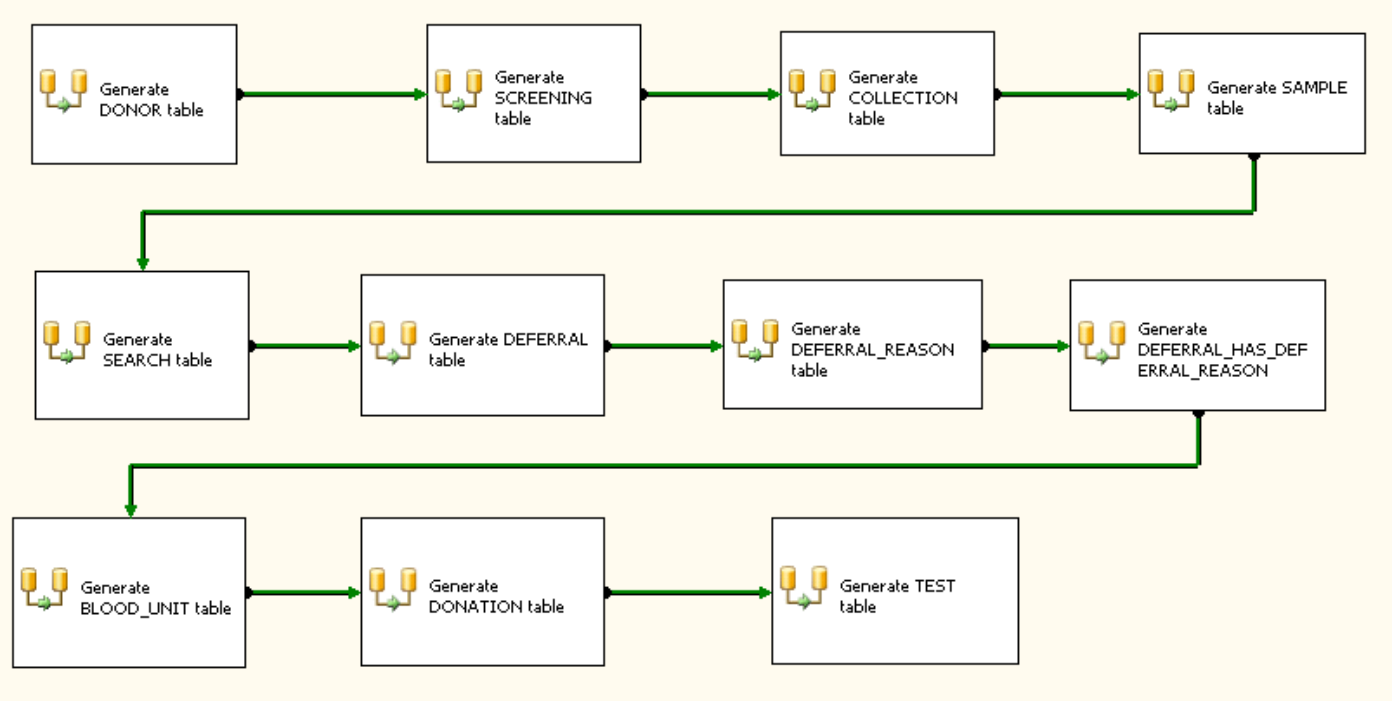


Figura 17: *Rotina "Populate Temporary NDS".*

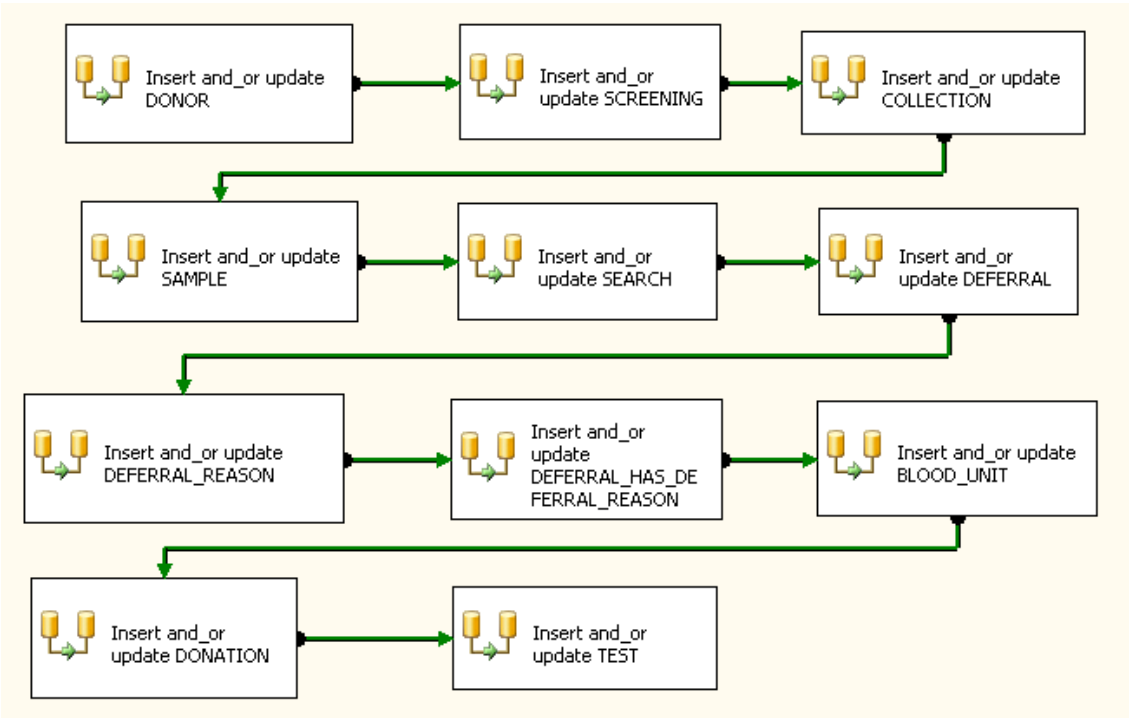


Figura 18: *Rotina "Upsert NDS".*

7 Conclusão

7.1 Procedência de dados

Neste trabalho conseguimos mostrar que, se é possível verificar a procedência dos dados, é possível qualificá-los, ou seja, apontar quais os problemas mais comuns e como corrigí-los. A partir daí é possível elaborar um bom modelo conceitual o que implica em rotinas ETL que conseguem levar os dados a um estado confiável e que consequentemente levam a um DW melhor.

7.2 Ferramentas

Durante o desenvolvimento, conforme as rotinas ETL tornavam-se maiores e mais complexas, percebia-se que o sistema ficava cada vez mais difícil de ser alterado. Quando a implementação teve fim, verificou-se que boa parte dessa dificuldade era devido ao ambiente de desenvolvimento, que era visual. Em ambientes visuais, praticamente não há reaproveitamento de código, uma vez que não há código, e sim componentes configuráveis. Quando o pacote SQL Server 2008 foi escolhido, havia uma grande esperança de que essas configurações de componentes pudessem ser reaproveitadas utilizando, por exemplo, variáveis. No entanto, apesar das melhorias serem muito grandes, quando comparadas ao pacote SQL 2000, ainda não foi alcançada a flexidade necessária.

Note que, o pacote Pentaho, por também ser visual e por ser fortemente baseado nas soluções Microsoft sofre dos mesmos problemas.

7.3 O Futuro

Por ora, o objetivo é procurar por soluções não visuais que substituam partes do processamento. Já foram encontrados alguns arcabouços de código aberto para desenvolvimento de rotinas ETL que se integram com o SQL Server 2008. Entretanto, o grupo de BD estuda seriamente o desenvolvimento de sua própria solução não visual para desenvolvimento de rotinas, pois, aparentemente, essa é única forma de que todas as suas necessidades sejam atendidas.

8 Parte Subjetiva

8.1 Desafios/Frustrações

8.1.1 Desafios

Desenvolver um projeto tão grande individualmente em cinco meses sem que as atividades paralelas da Iniciação Científica, da graduação e pessoais fossem interrompidas foi o maior desafio. Porém, desde o momento da escolha do projeto, o aluno de doutorado Pedro L. Takecian sempre ofereceu grande ajuda e se demonstrou interessado no meu trabalho. Em especial, pode-se citar o projeto do Modelo Conceitual e a implementação das rotinas ETL onde a experiência dele de 4 anos com os dados da Fundação Pró-Sangue foi crucial para preencher as lacunas dos meus 2 anos de experiência.

8.1.2 Frustrações

Cada vez que, devido ao tempo escasso nos vemos obrigados a retirar uma funcionalidade ou objetivo inicial, temos uma frustração. Entretanto, a maior frustração ocorreu no momento em que percebemos que, apesar de todas as funcionalidades promissoras do pacote SQL Server 2008, as rotinas ETL que estavam sendo implementadas demonstravam problemas muito parecidos aos encontrados nas implementações anteriores. Conseguimos resolver muitos deles, no entanto, o reutilização de código ainda é precária e isso, certamente vai nos obrigar, futuramente, a repensar a utilização de soluções visuais.

8.2 Disciplinas Importantes

MAC0323 - Estrutura de Dados - Foi a disciplina onde aprendi realmente a programar e onde nos foram ensinadas as primeiras estruturas de dados e como elas devem ser escolhidas com cuidado.

MAC0436 - Sistemas de Bancos de Dados - Foi a disciplina que desmitificou o conceito e o uso de bancos de dados os quais até então eram uma incógnita, visto que, nenhuma das matérias cursadas anteriormente sequer propôs seu uso em algum exercício-programa.

MAC0422 - Sistemas Operacionais - Essa disciplina foi muito interessante e, apesar de não se relacionar diretamente com o assunto desse trabalho, foi importante, sob o aspecto da pesquisa e "mãos na massa".

MAC0438 - Programação Concorrente - Os exercícios-programa dessa disciplina eram muito interessantes, no entanto, muito complicados porque além dos conceitos de programação concorrente, utilizavam também a linguagem Java. Foi nessa disciplina que tivemos a oportunidade de programar verdadeiramente utilizando essa linguagem de programação.

8.3 Planos futuros

8.3.1 Sobre o trabalho

Este projeto é apenas o ponto de partida de um projeto maior que visa integrar os quatro novos hemocentros que iniciaram parceria com o grupo de BD e o governo brasileiro. Como já foi

explicado, estamos avaliando seriamente a possibilidade de implementarmos nossa própria solução, uma vez que o projeto está tomando proporções que ferramentas visuais não têm condições manter. Além disso, temos procurado por arcabouçoes não visuais e de código aberto que permitam o desenvolvimento de rotinas ETL para integrar com as soluções visuais utilizadas hoje em dia. Talvez essa seja a solução menos custosa, no entanto é necessário saber quais limitações essa solução impõe ao sistema.

8.3.2 Sobre a vida acadêmica

Recentemente recebi a notícia de que fui um dos aprovados para participar de um intercâmbio em uma universidade francesa no próximo semestre, então, no momento, meus planos são, unicamente melhorar o meu francês e me preparar para a viagem. Quando eu voltar, gostaria de continuar trabalhando com o Prof. João Eduardo nos projetos do grupo de BD e iniciar um mestrado na área de Inteligência de Negócios, Procedência e Qualidade de dados ou na área de Transações.

9 Referências

Referências

[1] Elmasri, Ramez e Navathe, Shamkant B.: *Fundamentals of Database Systems, Fourth Edition*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003, ISBN 0321122267.

[2] Group, PostgreSQL Global Development: *PostgreSQL*. `http://www.postgresql.org/`, Acessado Novembro/2009.

[3] J. E. Ferreira, I. C. Italiano, O. K. Takai: *Introdução a banco de dados*, 2005. `http://www.ime.usp.br/~jef/apostila.pdf`, Acessado Novembro/2009.

[4] JPivot: *JPivot*. `http://jpivot.sourceforge.net/`, Acessado Novembro/2009.

[5] KIMBALL, R. et al.: *Introducing data warehouse architecture*. Wiley Computer Publishing, 1998.

[6] Microsoft: *Microsoft Developer Network (MSDN)*. http://msdn.microsoft.com/en-us/library/default.aspx, Acessado Novembro/2009.

[7] Microsoft: *Sql Server 2008*. `http://www.Microsoft.com/SQLServer`, Acessado Novembro/2009.

[8] Microsoft: *Visual C# Developer Center*. http://msdn.microsoft.com/pt-br/vcsharp/default.aspx, Acessado Novembro/2009.

[9] Microsystems, Sun: *Developer Resources for Java Technology*. `http://java.sun.com/`, Acessado Novembro/2009.

[10] OIKAWA, M. K., SILVA, P. P. S. B., MERINO, E. F., FERNANDEZ-BECERRA, C., WUNDERLICH, G., BARRERA, J., DELPORTILLO, H. A., e FERREIRA, J. E.: *ClinMaldb: a clinical field-research oriented relational database to study human malaria*. In *IADIS International Conference e-Health 2009*, 2009.

[11] Pentaho: *Mondrian OLAP Server*. `http://mondrian.pentaho.org/`, Acessado Novembro/2009.

[12] Pentaho: *Pentaho Data Integration*. http://www.pentaho.com/products/data_integration/, Acessado Novembro/2009.

[13] Rainardi, Vincent: *Building a data warehouse: with examples in SQL Server*. APRESS, 1 edição, 2008, ISBN 1-59059-931-4.

[14] Wikipédia: *Data warehouse*. http://en.wikipedia.org/wiki/Data_warehouse, Acessado Novembro/2009.

[15] Wikipédia: *Workflow*. `http://en.wikipedia.org/wiki/Workflow`, Acessado Novembro/2009.