O PROCESSO DE ETL NA CONSTRUÇÃO DE CONHECIMENTO EM UMA APLICAÇÃO DE UMA EMPRESA SEGURADORA

MARCELO AUGUSTO SOUZA DA COSTA

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação
Orientador: Prof. Tarcísio de Souza Lima

JUIZ DE FORA, MG - BRASIL
DEZEMBRO, 2009

O PROCESSO DE ETL NA CONSTRUÇÃO DE CONHECIMENTO EM UMA APLICAÇÃO DE UMA EMPRESA SEGURADORA

Marcelo Augusto Souza da Costa

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DA UFJF – UNIVERSIDADE FEDERAL DE JUIZ DE FORA COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

_____

Prof. Tarcísio de Souza Lima – orientador
MSc. em Informática, PUC-RJ, 1988

_____

Profa. Lúcia Helena de Magalhães – DCC/UFJF
Msc. em Sistemas Computacionais, COPPE-Civil/UFRJ, 2008

_____

Profa. Tatiane Ornelas Martins Alves
Esp. em Ciência da Computação, UFV, 2004

JUIZ DE FORA, MG - BRASIL

DEZEMBRO, 2009

SUMÁRIO

LISTA DE FIGURAS

LISTA DE REDUÇÕES

DW	*Data Warehouse*
ETL	*Extraction, Transformation and Load*
KDD	*Knowledge Discovery in Databases*
OLAP	On-Line Analitycal Process
OLTP	*Online Transaction Processing*
SGBD	Sistema Gerenciador de Bando de Dados

RESUMO

Com o avanço da tecnologia, as bases de dados das empresas passaram a acumular grandes volumes de dados. Além deste problema tem, também, o problema de inconsistência dos dados e de descentralização de suas bases. Neste contexto surgem estratégias para auxiliar no processo de tomadas de decisão. Dentre essas estratégias, destacam-se o processo de KDD (*Knowledge Discovery in Databases*) e de DW (*Data Warehouse*). Para que o desenvolvimento do processo de KDD e do DW seja feito com sucesso é precisso realizar um tratamento nos dados das bases utilizadas. Este tratamento é conhecido como ETL (*Extraction, Transformation and Load*) e consiste em extrair os dados dos bancos de dados, realizar um processo de limpeza e transformação e, então, realizar a carga. Este é o foco principal deste trabalho, que além de apresentar os princiais conceitos de DW e KDD é feito um processo prático de ETL utilizando uma ferramenta própria para isto.

Palavras-chave:

Data Warehouse, *Knowledge Discovery in Databases*, ETL

CAPÍTULO 1
INTRODUÇÃO

Um dos maiores bens que as empresas possuem são os dados sobre suas transações. Em suas bases de dados têm-se os registros das aplicações realizadas em seu cotidiano e através destes é possível obter informações que são muito importantes para gerar informação que pode auxiliar a reduzir custos, aumentar lucros, descobrir novas oportunidades de mercado ou qualquer outra estratégia que uma empresa possa vir a tomar (FARO, 2002). Para que uma instituição utilize seus dados de forma eficaz, duas opções se destacam, que são o DW (*Data Warehouse*) e o KDD (*Knowledge Discovery in Databases*).

O DW, termo utilizado primeiramente por INMON em 1990 (MELO, 2000), significa "armazém de dados" e pode ser definido como um tipo de banco de dados especial destinado a sistemas de apoio à decisão e cujos dados são armazenados em estruturas lógicas dimensionais possibilitando o seu processamento analítico por ferramentas especiais (BARBIERI, 2001). Uma das principais missões do DW é contribuir de forma significante para o processo de decisão da organização. O sucesso deste objetivo começa e termina com seus usuários, pois, além deles definirem o que deve ser carregado no DW, são eles que utilizam o sistema (GONÇALVES, 2003).

O KDD é um processo capaz de descobrir conhecimento em bancos de dados. Segundo FAYYAD, SHAPIRO e SMYTH (1996b), este processo foi proposto em 1989 para referir-se às etapas que produzem conhecimentos a partir dos dados. Dentro deste processo a etapa de mineração de dados é a fase que transforma dados em informação. Seu objetivo principal é extrair conhecimento a partir de grandes bases de dados.

Tanto o DW quanto o KDD utilizam em sua fase inicial um processo conhecido como **ETL (*Extraction, Transformation and Load*)**[1]. Este processo é responsável por extrair os dados dos sistemas operacionais de origem, transformá-los para deixá-los consistentes e carregá-los no DW (OLIVEIRA, 2002). O processo de ETL é a parte mais custosa na construção do DW e no processo de KDD, consumindo cerca de 70% dos recursos de sua implementação e manutenção. Devido a esta grande importância e ao fato de existir pouco material bibliográfico em língua portuguesa que trata deste assunto de uma forma mais

[1] Em português, Extração, Transformação e Carga

detalhada, tanto em material eletrônico quanto em material impresso, o processo de ETL é o assunto principal deste trabalho.

Os principais objetivos do processo ETL são (KIMBALL, 2004):

- Remover erros e corrigir dados faltantes;
- Assegurar a qualidade dos dados;
- Capturar o fluxo de dados transacionais;
- Ajustar dados de múltiplas origens e usá-los juntos;
- Fornecer estruturas de dados para serem utilizadas por ferramentas pelos analistas responsáveis pelo desenvolvimento;
- Fornecer dados em formato físico para serem usados por ferramentas dos usuários finais.

Um processo ETL pode ser feito de forma manual ou utilizando ferramentas próprias para tal. O processo feito por ferramentas possuem mais vantagens em relação ao processo manual: desenvolvimento mais rápido e simples, não é necessário ter profundos conhecimentos em programação, as ferramentas ETL geram automaticamente os metadados, apresentam estágios nativos de conexão a banco de dados, entre outras. As ferramentas têm um nível de aprendizado acentuado, mas mesmo sendo uma ferramenta de difícil utilização, que exige investimentos em pessoal, são compensados com o desempenho e flexibilidade da mesma (OLIVEIRA, 2002).

1.1 Objetivo da Monografia

Este trabalho tem por objetivo explorar com detalhes e apresentar de forma prática o processo de ETL. Para isto são conceituados o processo de KDD – *Knowledge Discovery in Databases*[2], o *Data Warehouse* (DW) e o ETL, que é um ponto fundamental para alcançar o objetivo dos dois primeiros.

Através de um estudo de caso de uma seguradora, utilizando uma ferramenta especializada em ETL, é demonstrado de forma prática como é feito o processo ETL. Esta prática não visa a construção de um DW ou realizar o processo de KDD, mas simular projetos ETL que são encontrados em empresas. Como o objetivo é demonstrar o processo ETL, nenhuma análise é feita sobre os arquivos gerados.

[2] Descoberta do Conhecimento em Base de Dados

1.2 Metodologia Utilizada

Foram feitos uma pesquisa e um levantamento de diversos artigos e livros relacionados aos assuntos de ETL, DW e KDD. Depois de expostos os objetivos destes três assuntos, uma análise mais profunda é feita em cima do processo ETL.

Após descritos estes processos, foi feito um estudo teórico sobre as ferramentas ETL disponíveis no mercado. Depois deste estudo, uma ferramenta foi selecionada e foi estudado as suas principais características. Além destas características, em um apêndice, é explicado os principais estágios que compõem esta ferramenta, para um melhor entendimento da parte prática.

Em seguida, foi selecionada uma base de dados de uma seguradora e realizado de forma prática o processo de ETL. O processo completo de ETL deste projeto envolve a geração de cerca de 50 arquivos, porém serão expostos 2 processos, além de uma preparação anterior e posterior a estes processos.

1.3 Estrutura da Monografia

Este trabalho visa apresentar de forma prática um processo ETL que é comumente encontrado em empresas. Desta forma, os capítulos foram divididos de acordo com os assuntos pertinentes à realização do trabalho, sendo no total, além desta introdução, quatro capítulos e mais o apêndice.

No segundo capítulo são apresentados os principais conceitos sobre *Data Warehouse.* No terceiro capítulo é feito um estudo mais detalhado sobre o processo ETL, além de uma análise das principais ferramentas encontradas no mercado. Dentre essas ferramentas, foi escolhida uma das ferramentas líderes de mercado e feito uma introdução sobre suas principais características. No quarto capítulo é feito um estudo de caso. Para este estudo de caso, foi selecionado um projeto ETL de seguradora, e através de sua base de dados é utilizado a ferramenta selecionada no terceiro capítulo para realizar o processo de extração, transformação e carga dos dados. O quinto capítulo conclui este trabalho, apresentando o conhecimento adquirido, limitações e dificuldades enfrentadas e sinaliza possíveis trabalhos futuros.

Depois do quinto capítulo são registradas as referências bibliográficas utilizadas para o desenvolvimento do trabalho. Por fim, um apêndice apresenta os principais estágios que compõem a ferramenta, a fim de facilitar o entendimento do estudo de caso.

CAPÍTULO 2
ETL NO CONTEXTO DE KDD E DW

Antigamente, o armazenamento de dados por pessoas e empresas era feito por anotações em papéis e as organizações tinham inúmeros arquivos de dados para cada aplicação específica. Com o tempo, a tecnologia evoluiu e as empresas puderam automatizar e armazenar os dados em SGBDs (Sistemas Gerenciadores de Banco de Dados), criando assim uma organização mais eficiente e eficaz dos dados (FARO, 2005). Atualmente, os bancos de dados das empresas recebem muitos registros que são essenciais para o seu funcionamento e organização. Com esse aumento de dados armazenados pelas empresas, as pesquisas e consultas em seus bancos tornaram-se uma tarefa difícil de ser realizada, pois além da quantidade de dados, estes podem estar alocados em várias bases de dados diferentes e a forma com que são armazenados, muitas vezes, não possuem um mesmo padrão para todos os bancos de dados da empresa, tornando-os inconsistentes.

Com isso é impraticável realizar uma interpretação e análise manual, pois são lentas e caras (FAYYAD, SHAPIRO e SMYTH, 1996a). Esses problemas podem fazer com que um profissional da área de negócios tome decisões baseadas em consultas pouco eficazes ou simplesmente de forma intuitiva que podem gerar prejuízos e perda de tempo. Com isso é preciso criar técnicas e ferramentas com objetivo de auxiliar os profissionais analistas, de uma forma eficaz e automática, na procura de informações estratégicas nos dados (BOSCARIOLI, 2002). Neste contexto de técnicas e ferramentas estão inseridos o KDD (*Knowledge Discovery in Databases*) e o DW (*Data Warehouse*).

2.1 O Processo de KDD

O KDD, cuja tradução revela seu objetivo - descoberta do conhecimento em banco de dados – é um processo para identificar padrões válidos, novos, potencialmente úteis e compreensíveis em dados para gerar conhecimento e auxiliar na tomada de decisão (FAYYAD, SHAPIRO e SMYTH, 1996a). O *KDD* é um processo não trivial que envolve várias etapas distintas e iterativas, até que seu objetivo seja atingido (FAYYAD, SHAPIRO e SMYTH, 1996a). Estas etapas são mostradas na figura 2.1.

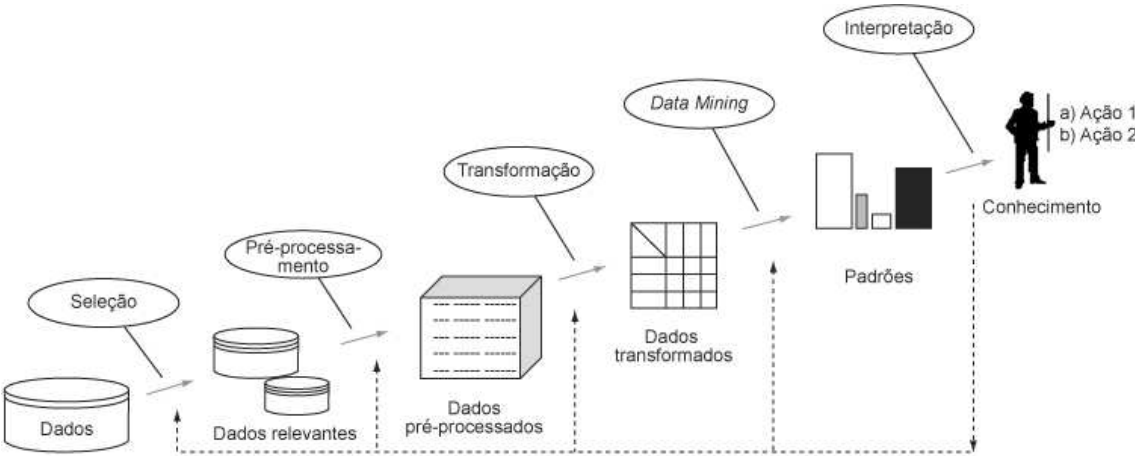


Figura 2.1 – Etapas do processo de KDD (adaptada de FAYYAD, SHAPIRO e SMYTH, 1996)

Na primeira etapa do processo de KDD – Seleção – é preciso ter o entendimento do domínio da aplicação. Ainda nesta etapa é preciso criar um conjunto de dados alvo na qual é feito a descoberta.

Em seguida, na segunda etapa – Pré-Processamento – são feitas operações básicas como remoção de dados inapropriados, criação de estratégia para tratamento de campos faltantes, eliminação de registros repetidos.

Na etapa de transformação é onde é feito o tratamento dos dados de acordo com o objetivo. Esta é uma das principais etapas, pois todo o processo depende que os dados tenham o mínimo de inconsistência possível.

A quarta etapa é o processo de *Data mining*. É nesta etapa que é feito a pesquisa e descoberta dos padrões. Para realizar esta etapa é preciso escolher os algoritmos para serem usados nas pesquisas por padrões em dados, como decisões de quais modelos e parâmetros são mais apropriados

A última etapa é a parte de interpretação dos padrões descobertos e possibilidade de retorno para qualquer das etapas anteriores, tão bem quanto possível visualização dos padrões extraídos, remoção de padrões redundantes ou irrelevantes, e tradução dos termos úteis em termos entendíveis pelos usuários.

As três primeiras etapas do KDD: seleção, pré-processamento e transformação é o alvo deste trabalho, o que é discutido com mais detalhes no capítulo 4.

2.2 – *Data Warehouse*

Com o *Data Warehouse* (DW), os usuários de negócio criam estratégias, através do acesso aos dados com uma maior qualidade, ajudando a avaliar as atividades emergentes do negócio, abrangendo, assim, as necessidades do mundo dos negócios (KIMBALL, 2002). Existem, na literatura, algumas definições de *Data Warehouse.*

Para CHAUDHURI (1997), *Data Warehouse* é um banco de dados histórico, separado lógica e fisicamente do ambiente transacional da organização, concebido para armazenar dados extraídos deste ambiente.

Já para OLIVEIRA (2002), *Data Warehouse* é um ambiente especializado que filtra, integra e disponibiliza informações gerenciais a partir de sistemas operacionais e fontes externas.

Ao reunir informações dispersas pelos diversos sistemas de origem, o *Data Warehouse* permite que sejam feitas análises bastante eficazes, transformando dados esparsos em informações estratégicas, antes inacessíveis ou subaproveitadas (TAURION, 1998 *apud* MARTINS, 2003).

INMON (1999), considerado por muitos pesquisadores como o “pai” do DW, define *Data Warehouse* como um conjunto de dados orientado a assunto, integrado, não volátil e variável com o tempo, criado para dar suporte à decisão. Estas características são apresentadas abaixo.

Um DW é considerado orientado a assunto, pois ao projetar o DW, as informações são organizadas por assuntos de análise de negócio, como por exemplo, vendas, clientes, produtos enquanto que os sistemas de informações tradicionais são orientados a processo como estoques, entrada e saída de matérias (MELO, 2000). Ao realizar o projeto de um DW, os dados são agrupados por assunto de interesse da empresa e, portanto, as informações não relacionadas ao assunto escolhido deverão ser desconsideradas, pois não serão úteis ao processo de apoio à tomada de decisão (MELO, 2000).

A característica de ser integrado é o aspecto mais importante em um DW, e essa integração é a essência do ambiente (MELO, 2000). Ser integrado significa que os dados que possuem origem em bases diferentes são carregados em uma única base com as convenções dos dados formalmente unificadas através de técnicas apropriadas para assegurar a consistência das informações (MELO, 2000). A figura 2.2 mostra essa característica.

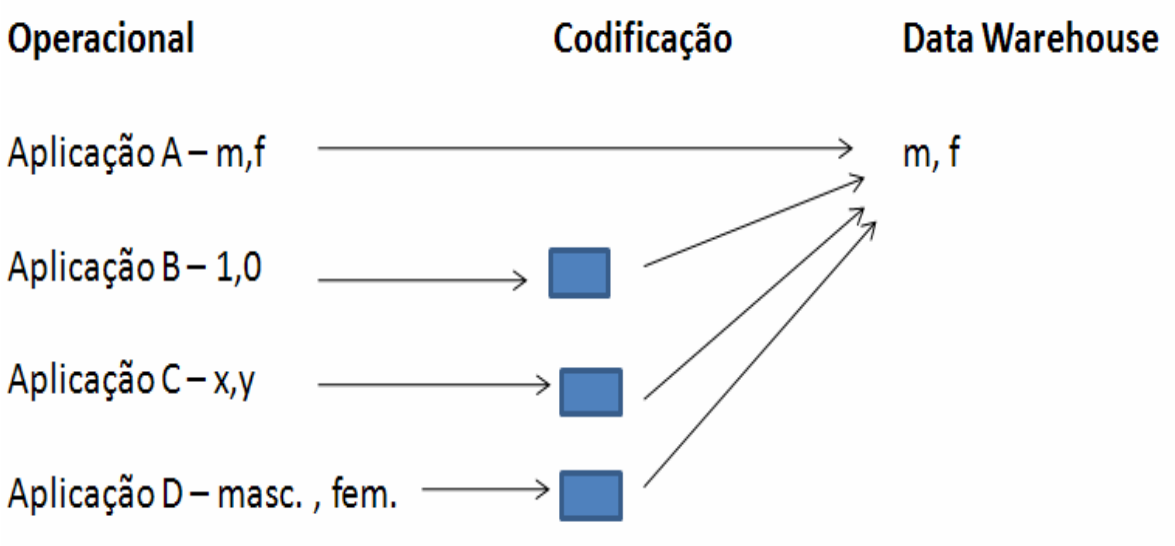


Figura 2.2 - Característica de Integração do DW (adaptada de CARDOSO, 2002)

A figura 2.2 apresenta quatro fontes de dados diferentes representadas por Aplicação A, Aplicação B, Aplicação C e Aplicação D, onde em cada uma delas há uma forma diferente de representar o atributo sexo. Estes diferentes formatos dos dados, presentes nas aplicações, são codificados para se transformar em um único formato padrão (“m” e “f” para masculino e feminino respectivamente) que é carregado no DW.

O DW é considerado não volátil pelo fato de não haver atualizações freqüentes nos dados que o povoam como acontece em banco de dados transacional (GONÇALVES, 2003). As alterações que são comuns de ocorrer em banco de dados transacionais não significa que é necessário alterar os dados no DW, mas sim gerar uma nova carga de dados. Estas cargas são realizadas de forma periódica com intuito de guardar históricos para servir de consultas (GONÇALVES, 2003).

A figura 2.3 faz uma comparação entre o sistema operacional e o *Data Warehouse* em relação a volatilidade. No sistema operacional o usuário pode executar operações de excluir, incluir, atualizar, substituir enquanto que no DW as operações comuns são a carga e o acesso aos dados.

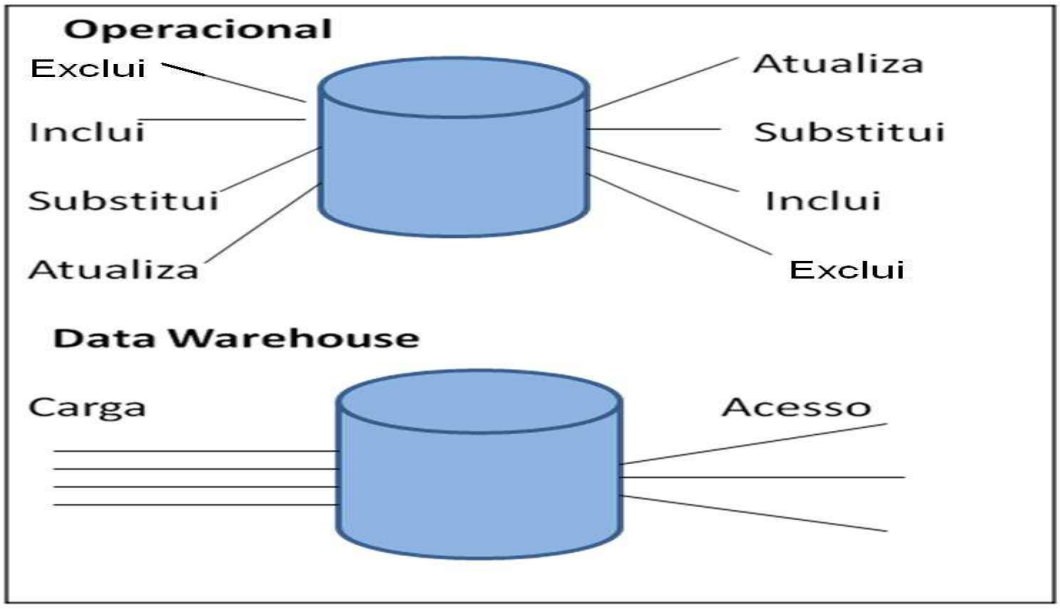


Figura 2.3 – Característica não volátil do *Data Warehouse* (adaptada de GONÇALVES, 2003)

A última característica do DW segundo INMON (1999) é ser considerado variante no tempo. Esta característica significa que as informações são armazenadas ao longo do tempo (GONÇALVES, 2003). Cada alteração nos dados implica uma nova carga de dados no DW. Com isto, os dados em um DW tornam-se conjuntos estáticos de dados, chamados de *snapshots*, de uma ou mais base de dados, capturados em certo momento que permite que os analistas de negócios visualizem as variações de uma determinada informação ao longo do tempo (GONÇALVES, 2003).

O DW possui vários objetivos e dentre eles os principais são:

- Fazer com que informações de uma empresa possam ser facilmente acessadas;
- Apresentar as informações da empresa de modo consistente;
- Ser adaptável e flexível a mudanças;
- Armazenar as informações de forma segura;
- Funcionar com uma base para melhor tomada de decisões.

2.2.1 Componentes de um *Data Warehouse*

Segundo KIMBALL (2004), um ambiente de DW é formado por quatro componentes separados e distintos: sistemas operacionais de origem, *data staging area*, área de apresentação dos dados e ferramentas de acesso a dados.

KIMBALL (2004) faz uma comparação destes componentes do DW com um restaurante, onde os clientes do restaurante são os usuários finais e a comida são os dados. Antes da comida ser servida, ela é preparada sob a supervisão de alguém com experiência.

Este profissional é o responsável pela parte ETL. Na cozinha, neste contexto é a *staging area*, onde a comida é selecionada, limpa, cortada, cozinhada e preparada para a apresentação final. A cozinha é a área de trabalho e está fora dos limites dos clientes. Se um cliente precisar de informação sobre como é preparada a comida, o *chef* deve apresentar ao cliente como a comida é feita. A figura 2.4 mostra estes quatro componentes.

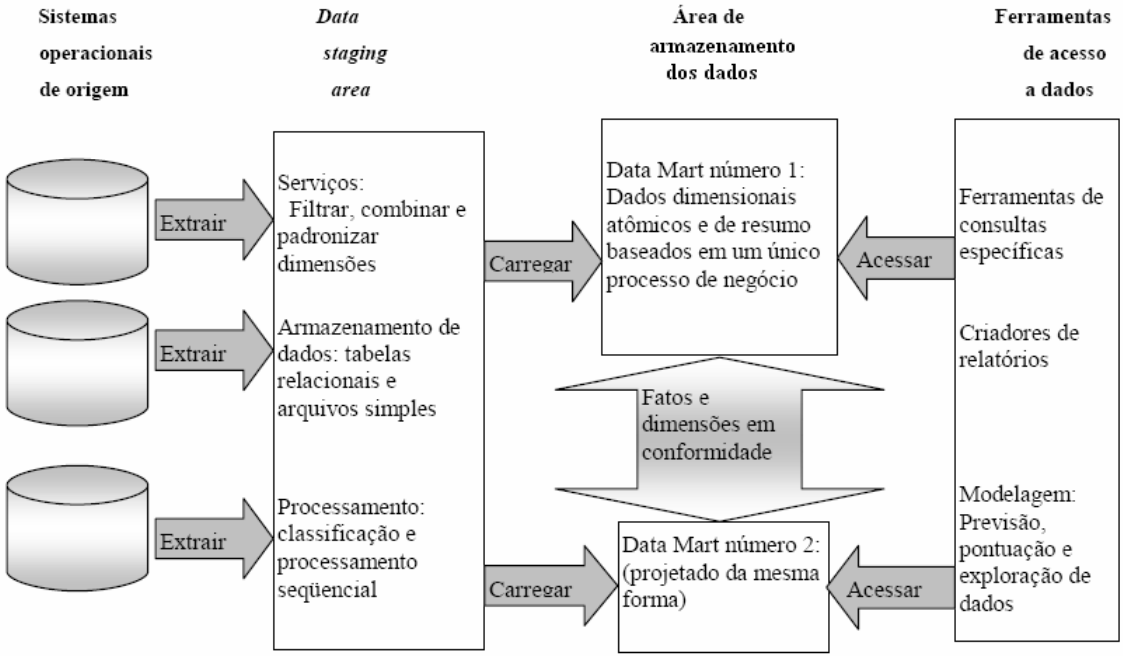


Figura 2.4 – Componentes de um *Data Warehouse* (adaptada de KIMBALL e ROSS, 2002)

Os sistemas operacionais de origem também conhecidos como OLTP *(Online Transaction Processing)* são os sistemas que compõem o ambiente operacional de uma empresa e realizam a captura das transações de negócio. Esses sistemas acumulam dados detalhados a partir das transações diárias da empresa (MELO, 2000). Na comparação de Kimball, esta área seria a dispensa do restaurante, onde ficam os alimentos.

Data staging area ou área de transporte de dados é onde se realiza o processo de extração, transformação e carga dos dados (KIMBALL e ROSS, 2002). Nesta etapa, os dados são capturados de forma bruta dos sistemas operacionais de origem, transformados e padronizados para, então, serem carregados na área de apresentação do *Data Warehouse* (KIMBALL e ROSS, 2002). Esta etapa é o assunto principal do trabalho e é apresentado com mais detalhes no capítulo 4.

Área de armazenamento dos dados é o local em que os dados ficam organizados, armazenados e tornam-se disponíveis para serem consultados diretamente pelos usuários, por criadores de relatórios e por outras aplicações de análise (KIMBALL e ROSS, 2002). A área

de armazenamento dos dados pode ser vista como uma série de *Data Marts* integrados, cada um representando um único processo de negócio (KIMBALL e ROSS, 2002).

As ferramentas de acesso a dados, conhecidas como OLAP, são utilizados pelos usuários da comunidade de negócios para acessar os dados. Com estas ferramentas os usuários de negócio conseguem recuperar informações pertinentes para que possam tomar decisões analíticas (KIMBALL e ROSS, 2002).

2.3 – Etapas de Seleção, Pré-Processamento e Transformação

Nas duas seções anteriores foram conceituados o processo de KDD e o DW. Nesta seção serão introduzidas as etapas mais custosas destes processos, antes de discutí-lo com mais detalhes no próximo capítulo. Estas etapas são: seleção, pré-processamento e transformação.

2.3.1 Seleção de Dados

Antes de iniciar o processo é importante, nestes dois casos, ter um amplo conhecimento do domínio da aplicação e dos objetivos. Com estes "pré-objetivos" alcançados é feito a seleção dos dados (CARVALHO *et al.*, 2002). A seleção dos dados visa obter uma massa de dados para atingir o objetivo. Para isto inicia-se verificando quais tabelas dos bancos de dados deverão ser utilizadas. Depois de definido as tabelas necessárias, é preciso analisar quais os atributos são necessários para a realização dos processos. Algumas bases de dados podem sofrer atualizações diárias, enquanto outras recebem atualizações mensais (CARVALHO *et al.*, 2002). Com tudo isto, realizar a união destes dados em uma base de dados centralizada pode se tornar uma tarefa complexa, já que pode envolver dados de diversos SGBD diferentes utilizados por vários setores de uma instituição.

2.3.2 Pré-processamento e limpeza

O pré-processamento é feito depois de selecionados os dados necessários, pois devido aos problemas expostos na seção anterior a qualidade dos dados pode variar consideravelmente (CARVALHO *et al.*, 2002). Nesta etapa é feito o tratamento de ausências de dados, eliminação de dados incompletos, repetição de registros, problemas de tipagem, tratamento de dados inconsistentes (CARVALHO *et al.*, 2002). Os tratamentos destes relevantes problemas na qualidade de dados podem ocorrer devido a omissões na entrada de dados, problemas na conversão entre bases de dados, falhas em mecanismos de leitura, entre outros (BOSCARIOLI, 2002). O tratamento consiste em, tipicamente, substituir os valores ausentes, incompletos ou inconsistentes por valores *default*, substituí-los por um valor médio ou simplesmente excluí-los (BOSCARIOLI, 2002).

2.3.3 Transformação

Transformação é a etapa onde os dados sofrem uma determinada transformação, para que os dados fiquem em um formato padrão (BOSCARIOLI, 2002). Como exemplo de transformação tem-se a conversão de valores quantitativos em valores categóricos, ou seja, cada valor equivale a uma faixa. Idade entre 0 e 18 equivale a Faixa 1; idade entre 19 e 21 equivale a Faixa 2 e assim por diante.

As vantagens de transformar um atributo são: melhorar a compreensão do conhecimento descoberto, reduzir o tempo de processamento, diminuir o espaço de busca e facilitar os processos de tomada de decisão (BOSCARIOLI, 2002).

Estas três etapas fazem parte do processo de ETL, assunto mais elaboradamente discutido no próximo capítulo.

CAPÍTULO 3
O PROCESSO DE EXTRAÇÃO, TRANSFORMAÇÃO E CARGA

O processo de ETL é a principal etapa na construção de um *Data Warehouse* (KIMBALL, 2004). Este processo é o responsável por extrair os dados dos sistemas de origem, realizar a limpeza e transformações necessárias e carregar nas tabelas para que possam ser usadas na tomada de decisão (KIMBALL, 2004). O ETL é a etapa mais demorada na implantação de um DW, consumindo cerca de 70% do tempo de sua implantação (KIMBALL, 2004). O processo de ETL é composto de três etapas: extração, transformação e carga dos dados (KIMBALL, 2002). Estas etapas são detalhadas nas seções seguintes.

3.1 Extração de dados

A extração dos dados dos sistemas de origem é o primeiro passo do processo ETL (KIMBALL, 2004). Cada origem dos dados pode estar em um banco de dados diferente ou em plataformas diferentes (KIMBALL, 2004). Ao realizar o processo de extração é preciso ter bem definido todo o mapeamento das tabelas que serão utilizadas para fazer a extração e as tabelas em que será feita a carga (KIMBALL, 2004).

As várias alternativas para extração permitem balancear desempenho, restrições de tempo e de armazenamento (GONÇALVES, 2005). Por exemplo, se a fonte for um banco de dados *on-line*, pode-se submeter uma consulta diretamente ao banco para criar os arquivos de extração (GONÇALVES, 2005). O desempenho das aplicações ligadas às fontes pode cair consideravelmente se transações *on-line* e as consultas para extração competirem entre si (GONÇALVES, 2005). Para resolver este problema pode-se criar uma cópia corrente dos dados das fontes a partir da qual será feita a extração, mas isto faz com que tenha que ter um espaço maior em disco para armazenar os arquivos gerados (GONÇALVES, 2005)

A seleção dos dados do ambiente operacional pode ser complexa, pois é necessário várias vezes selecionar vários campos do sistema transacional para compor um único campo no DW, com isto é preciso o desenvolvimento de *queries*, que muitas vezes podem ser complexas e terem que ser feitas manualmente (OLIVEIRA, 2002).

A extração dos dados deve ser feita levando em consideração os objetivos do DW, pois a extração de dados desnecessários, além de ocupar um espaço considerável em disco, eleva o tempo de extração.

Um outro problema encontrado é que a documentação e o modelo de dados do ambiente operacional, muita das vezes não existem ou não estão documentados, o que exige do profissional, responsável por essa etapa, um grande esforço para compreender o sistema e analisar quais os dados de quais tabelas deverão ser extraídos (OLIVEIRA, 2002).

As rotinas de extração devem ser capazes de isolar somente os dados que foram inseridos e atualizados desde a última extração, sendo este processo conhecido como *refresh* (GONÇALVES, 2005). A melhor política de *refresh* deve ser avaliada pelo administrador do DW, que deve levar em conta características como as necessidades dos usuários finais, tráfego na rede e períodos de menor sobrecarga (GONÇALVES, 2005).

Depois de ter bem definido e planejado o processo de extração, o passo seguinte é a transformação dos dados, que é apresentada na seção seguinte.

3.2 Transformação dos dados

A transformação dos dados é o segundo e principal passo do processo ETL (KIMBALL, 2004). Depois de extraídos, os dados são copiados para *staging area*, onde é realizado o processo de transformação dos dados, através de uma série de tratamentos (GONÇALVES, 2005).

O primeiro destes tratamentos refere-se à limpeza dos dados ou filtragem dos dados, cujo objetivo é garantir a integridade dos dados através de programas ou rotinas especiais que tentam identificar anomalias e resolvê-las, deixando os dados em um estado consistente antes de estarem disponíveis no DW (GONÇALVES, 2005). Como exemplos de limpeza de dados tem-se: a correção de erros de digitação, a descoberta de violações de integridade, a substituição de caracteres desconhecidos, a padronização de abreviações (GONÇALVES, 2005).

O passo seguinte é colocar os dados em uma forma homogênea, aplicando uma metodologia de comparação de representações, que inclui os critérios a serem utilizados na identificação de semelhanças e conflitos de modelagem (GONÇALVES, 2005). Conflitos de modelagem podem ser divididos em dois: semânticos e estruturais (GONÇALVES, 2005). Os conflitos semânticos são todos aqueles que envolvem o nome ou palavra associada às estruturas de modelagem, como por exemplo, mesmo nome para diferentes entidades ou diferentes nomes para a mesma entidade (GONÇALVES, 2005). Já os conflitos estruturais

englobam os conflitos relativos às estruturas de modelagem escolhidas, tanto no nível de estrutura propriamente dito como no nível de domínios (GONÇALVES, 2005). Os principais tipos de conflito estruturais são os conflitos de domínio de atributo que se caracterizam pelo uso de diferentes tipos de dados para os mesmos campos (GONÇALVES, 2005). Conflitos típicos de domínio de atributo encontrados na prática são (GONÇALVES, 2005):

- Diferenças de unidades: quando as unidades utilizadas diferem, embora forneçam a mesma informação. Exemplo: distância em metros ou quilômetros.
- Diferenças de precisão: quando a precisão escolhida varia de um ambiente para outro. Exemplo: quando o custo do produto é armazenado com duas posições ou com seis posições decimais.
- Diferenças em códigos ou expressões: quando o código utilizado difere um do outro. Exemplo: sexo representado por M ou F e por 1 ou 2.
- Diferenças de granularidade: quando os critérios associados a uma informação, embora utilizando uma mesma unidade, são distintos. Exemplo: quando horas trabalhadas correspondem às horas trabalhadas na semana ou às horas trabalhadas no mês.
- Diferenças de abstração: quando a forma de estruturar uma mesma informação segue critérios diferentes. Exemplo: endereço armazenado em um atributo único ou subdividido em rua e complemento).

Outros tipos de modificações nos dados podem ser desejadas, tais como (OLIVEIRA, 2002):

- Realizar somatório por um determinado campo;
- Selecionar o primeiro ou o último registro de uma determinada coluna;
- Selecionar o valor máximo ou mínimo de uma coluna;
- Fazer a média dos registros de uma coluna;
- Ordenar os dados por uma coluna ou mais;

Depois de realizada a filtragem e transformação dos dados, passa-se para a etapa de carga dos dados. Esta etapa é detalhada na seção seguinte.

3.3 Carga dos dados

A etapa de carga dos dados é a última etapa do processo ETL (OLIVEIRA, 2002). Esta etapa possui uma complexidade alta e alguns fatores devem ser levados em conta (OLIVEIRA, 2002):

- Integridade dos dados: ao realizar a carga é necessário checar os campos que são chaves estrangeiras com suas respectivas tabelas para certificar que seus dados estão de acordo com a tabela primária.
- Carga incremental ou a carga por cima dos dados: a carga incremental, normalmente, é feita em tabelas fatos, e a carga por cima dos dados é feita em tabelas dimensão onde o analista terá que excluir os dados existentes e inserir todos os dados novamente. Mas em alguns casos, poderá acontecer que as tabelas de dimensão terão que manter o histórico, então, o mesmo deverá ser mantido. A decisão para o tipo de carga a ser feita deve ser planejada com cuidado, pois a carga pode levar um tempo elevado.
- Os arquivos a serem gerados devem obedecer a mesma ordem das colunas que foram estipuladas para o ambiente DW.
- Criação de rotinas: apesar de ter ferramentas especializadas nesta etapa, muitas vezes é necessário criar rotinas de carga para atender determinadas situações que poderão ocorrer.

Para a realização do processo ETL, existem no mercado diversas ferramentas próprias. Na próxima seção é feito um estudo sobre as ferramentas de ETL.

3.4 Ferramentas de ETL

Ferramentas de ETL são apropriadas para integração de dados que consistem de sincronizações de dados entre aplicações e processos interativos simples, projetos de integração de dados *on-line* que envolvam grandes quantidades de dados, transformações complexas ou incorporações de novos dados (OLIVEIRA, 2002).

As ferramentas são orientadas a extrair dados das tabelas relacionais, entendendo o significado das relações entre as tabelas, combinando e acrescentando dados oriundos de outras fontes. Isto pode envolver um *join* simples entre duas tabelas relacionais, ou *joins* complexos e heterogêneos envolvendo múltiplas tabelas de diferentes aplicações. Pode,

também, envolver transformações extremamente complexas (OLIVEIRA, 2002). Nestes casos, o usuário projeta o *job* na interface gráfica de desenvolvimento de fluxos da ferramenta de ETL, o que agiliza o processo, pois não é necessário escrever linhas de código (GONÇALVES, 2003).

As ferramentas de ETL oferecem um bom desempenho para movimentar e transformar dados, executando operações de bancos de dados em grandes volumes de dados (OLIVEIRA, 2002).

As principais ferramentas de ETL são apresentadas pelo Quadrante Mágico de Gartner[3], que é uma representação gráfica do mercado em um certo período de tempo (figura 3.1). O quadrante descreve as análises de Gartner sobre o desempenho de certos fornecedores de acordo com critérios do mercado (ROSADO, 2008). O Quadrante Mágico serve apenas como uma forma de pesquisa, e não como um guia específico de ações.

Gartner define como "líderes" os fornecedores que têm bom desempenho nos negócios, têm visão clara do mercado, e estão ativamente construindo competências para manter sua posição de liderança no mercado. "Visionários" são os fornecedores que demonstram uma visão das direções do mercado, que claramente estão focados na preparação para atender o mercado, mas que também podem ser criativos na oferta de serviços.

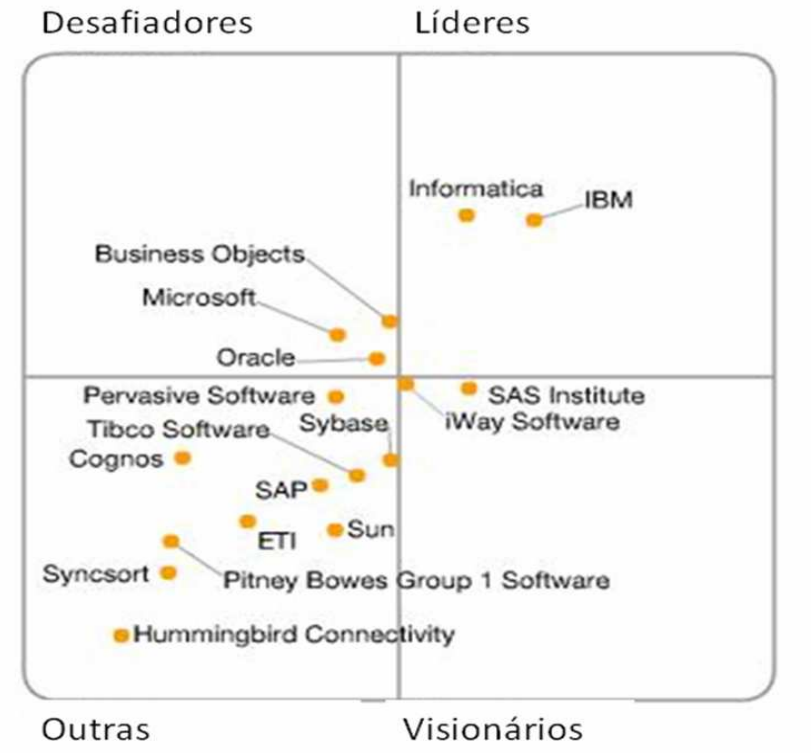


Figura 3.1 – Quadrante Mágico de Gartner

[3] http://mediaproducts.gartner.com/reprints/oracle/151150_0001.png (Nov, 2008)

Neste quadrante, na parte líderes estão as principais ferramentas de ETL do mercado: *DataStage da IBM* e *PowerCenter* da Informatica. Outras ferramentas com grande destaque são as ferramentas da Oracle (*Oracle Warehouse Builde),* da Microsoft (*SQL Server Integration Services),* e da *Business Objects* (*Business Object Data Integrator),* representadas no quadrante desafiadores.

3.4.1 Ferramenta de ETL *WebSphere DataStage*

A ferramenta *WebSphere DataStage* da IBM é uma das líderes e mais poderosas ferramentas de ETL do mercado. Esta ferramenta permite a integração sólida das informações corporativas.

A *WebSphere DataStage* oferece três recursos essenciais para uma integração de dados corporativos bem sucedida: uma abrangente conectividade, para um acesso fácil e rápido a qualquer sistema de origem ou destino; ferramentas avançadas de desenvolvimento e manutenção, que aceleram a implementação e simplificam a administração; e uma plataforma escalável que pode manipular com facilidade os volumes gigantescos de dados corporativos dos dias atuais.

O *DataStage* suporta a coleta, integração e transformação de grandes volumes de dados com estruturas variando de simples a altamente complexas, gerencia de forma rápida de pequenos e grandes volumes de dados. A ferramenta suporta um número virtualmente ilimitado de origens e destinos de dados heterogêneos em uma única tarefa, incluindo arquivos de texto, complexas estruturas de dados XML, sistemas de aplicativos corporativos que incluem SAP, Siebel, Oracle, quase todos os bancos de dados, inclusive bancos de dados particionados como o Oracle, IBM DB2 Universal Databa, Informix, Sybase, Teradata e Microsoft SQL Server, Web Services, SAS e outros.

O *DataStage* possui interface nativa para acesso a diversos bancos de dados relacionais, *Web services* e para aplicativos ERP, possui acesso baseado em padrões de indústria (ODBC, *Flat files*, FTP, XML), além de possuir bibliotecas internas com mais de 600 transformações para datas, *strings*, pesos, medidas. Além do grande número de funções presentes na ferramenta é possível desenvolver outras, para facilitar todo o processo.

Apesar de ser direcionada primeiramente para os ambientes de DW, o *DataStage* também pode ser utilizado em qualquer projeto de manipulação de dados, migração de dados ou projetos de reengenharia de informação.

A ferramenta *DataStage* é formada por 5 componentes:

- **DataStage *Administrator*:** Cria projetos de ETL, e define políticas de utilização.

- **DataStage *Designer*:** Desenvolve e mantém processos de ETL.
- **DataStage *Director*:** Executa, agenda e monitora processos de ETL criados pelo *DataStage* Designer.
- **DataStage *Manager*:** Manipula o repositório de metadados do *DataStage*, cria e mantém rotinas de transformação de dados.
- **DataStage *Server*:** Mantém o repositório de *metadados*, armazena os parâmetros de processos ETL, estabelece conexões com fontes e alvos de dados e realiza efetivamente o processo de extração, transformação e carga dos dados (Servidor).

3.4.1.1 O *DataStage Administrator*

O componente da ferramenta *DataStage Administrator* é utilizado para adicionar, remover ou configurar as propriedades de um projeto através de interface gráfica ou através de instruções diretas no repositório do *Universe*. Com ele é possível associar privilégios para grupos de usuários do ambiente com dois tipos de funções: *Operator* e *Developer*. Os usuários do grupo *Operator* podem executar *jobs* utilizando o componente *DataStage Director*, porém, não podem editá-los. Usuários do grupo *Developer* podem criar e editar *jobs.*

No *DataStage*, todos os processos de ETL são realizados e organizados em projetos. Os projetos são criados durante o processo de instalação ou adicionados e configurados pelo *DataStage Administrator*.

3.4.1.2 O *DataStage Manager*

O componente *DataStage Manager* oferece interface gráfica para a manipulação do repositório de metadados do *DataStage*, permitindo que a biblioteca de rotinas de transformação seja estendida utilizando codificação BASIC, onde depois de validadas, as mesmas são embutidas no repositório e disponibilizadas para todos os *jobs* no projeto.

Qualquer objeto do repositório em um projeto pode ser exportado para um arquivo e importado para outro projeto do *DataStage*. Este procedimento também é utilizado para a realização de *backups* de projetos.

3.4.1.3 O *DataStage Designer*

O *DataStage Designer* é responsável, através de visualização gráfica, pelo fluxo dos processos de ETL.

Um *Job* é uma unidade de execução que quando compilado, torna-se disponível para execução através do componente *DataStage Director* ou através de comandos de sistema. Os *jobs* são compostos por estágios e conectados através de ligações, chamados de *links*.

Um fluxo de dados é criado editando as propriedades dos estágios e ligações para realizar o processamento necessário.

3.4.1.4 O *DataStage Director*

O *DataStage Director* permite a validação, execução e monitoramento de *jobs* compilados pelo *DataStage Designer*.

Com o *DataStage Director* é possível agendar a execução de *jobs*, visualizar o *status* do *job* quanto a sua compilação, validação e execução. Também é possível visualizar o *log* de execução de cada *job*, facilitando assim a identificação de erros.

No capítulo seguinte é feito o processo de ETL utilizando a ferramenta *DataStage*.

CAPÍTULO 4
ESTUDO DE CASO: APLICAÇÃO DE ETL EM UMA EMPRESA SEGURADORA

Para exemplificar o processo de ETL apresentado anteriormente, neste capítulo será feito um estudo de caso. Este estudo de caso consiste na implementação de um projeto ETL em uma empresa de seguros. Este projeto tem o objetivo de substituir arquivos que são gerados por aplicações COBOL. Na aplicação original, programas desenvolvidos na linguagem COBOL geram arquivos com dados advindos de um banco de dados DB2 e estes arquivos são enviados para as filiais da seguradora para análise dos sinistros.

Esta seguradora irá substituir o sistema *mainframe* por um novo sistema desenvolvido em Java. O banco de dados utilizado neste novo sistema é o Oracle e a geração dos arquivos desejados é feito por um processo ETL utilizando a ferramenta IBM *DataStage*. O projeto completo gera cerca de cinqüenta arquivos. Para este estudo de caso será apresentado dois exemplos ETL, além da preparação antes e depois da geração destes arquivos.

Um projeto ETL é dividido em três etapas: desenvolvimento, homologação e produção. A figura 4.1 exemplifica estas etapas.

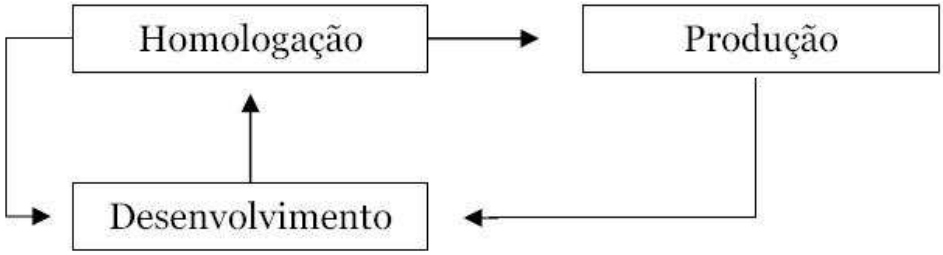


Figura 4.1 – Etapas do Projeto ETL

- Na etapa de desenvolvimento o analista de ETL tem liberdade para desenvolver, alterar, compilar, executar, gravar arquivos nos diretórios reservados para desenvolvimento e inserir ou atualizar dados na base de dados de desenvolvimento. Ao realizar os testes necessários, os *jobs* são passados para homologação.
- Na etapa de homologação o analista de ETL tem a liberdade apenas de compilar e executar os *jobs* desenvolvidos. Problemas nesta etapa devem ser resolvidos na etapa de desenvolvimento antes de passar para a etapa de produção.

- Na etapa de produção o analista de ETL consegue apenas visualizar o *log* de execução do job para verificar se sua execução ocorreu com sucesso. Se algum erro ocorrer, é necessário que se corrija na etapa de desenvolvimento.

Para realizar os processos de ETL é preciso ter uma especificação contendo os campos e as tabelas de origem, as regras de transformação e o formato dos campos do arquivo gerado. Esta especificação é feita com o auxílio dos usuários finais através de reuniões. Com a especificação pronta é preciso analisar e entender toda a especificação para que o desenvolvimento seja feito da forma mais otimizada possível para evitar que o processo demore muito para sua conclusão. Nas seções seguintes é mostrada a realização do processo de ETL.

4.1 O Processo de ETL – Importação de metadados

Para iniciar o processo de ETL é necessário ter os metadados das tabelas. Para obter os metadados das tabelas Oracle foi preciso solicitar ao DBA a criação de um nome de usuário e senha e a permissão deste para acesso as tabelas do banco de dados.

A importação dos metadados é feito pelo *DataStage Manager*, clicando em *Import*, selecionando o tipo de banco de dados desejado e preenchendo os requisitos de acesso a base. A figura 4.2 exemplifica esta importação.

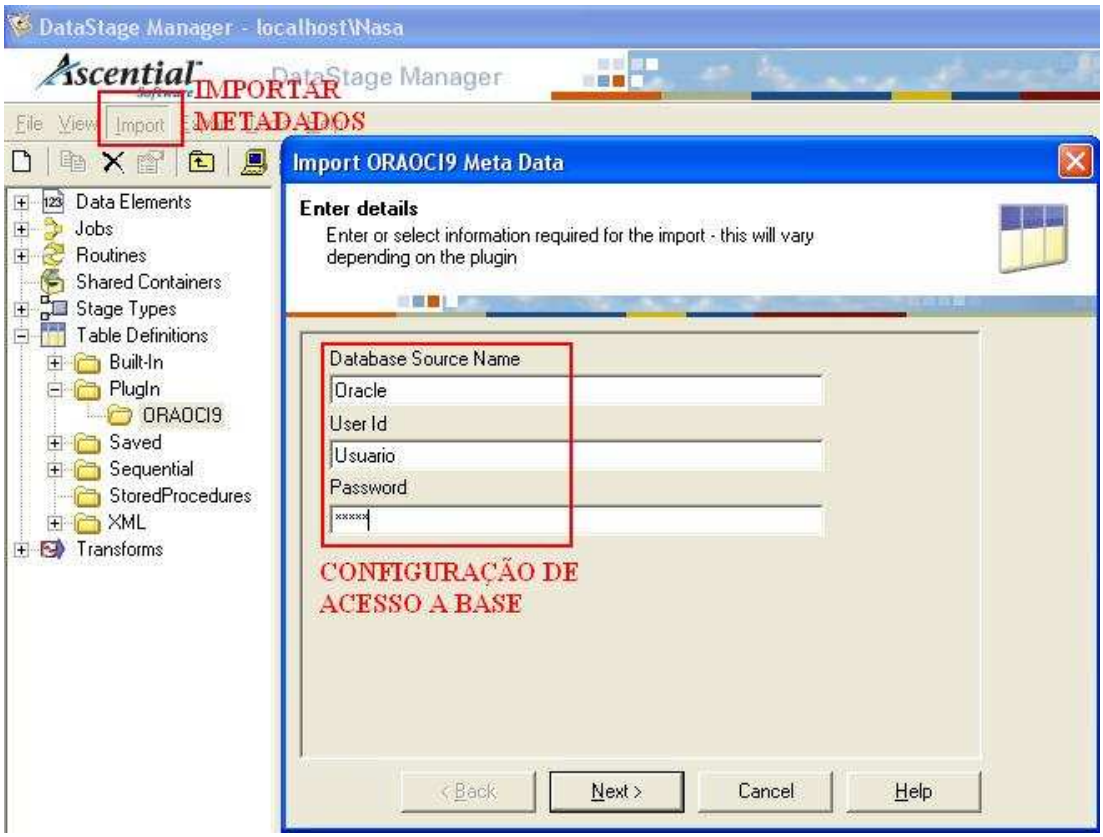


Figura 4.2 – Importação de metadados

A figura 4.3 apresenta uma amostra das tabelas importadas.

Figura 4.3 – Tabelas Importadas

A figura 4.4 mostra as colunas de uma tabela com seus metadados. É possível, ainda visualizar outras opções como as relações desta tabela com outras tabelas na aba *Relationships*.

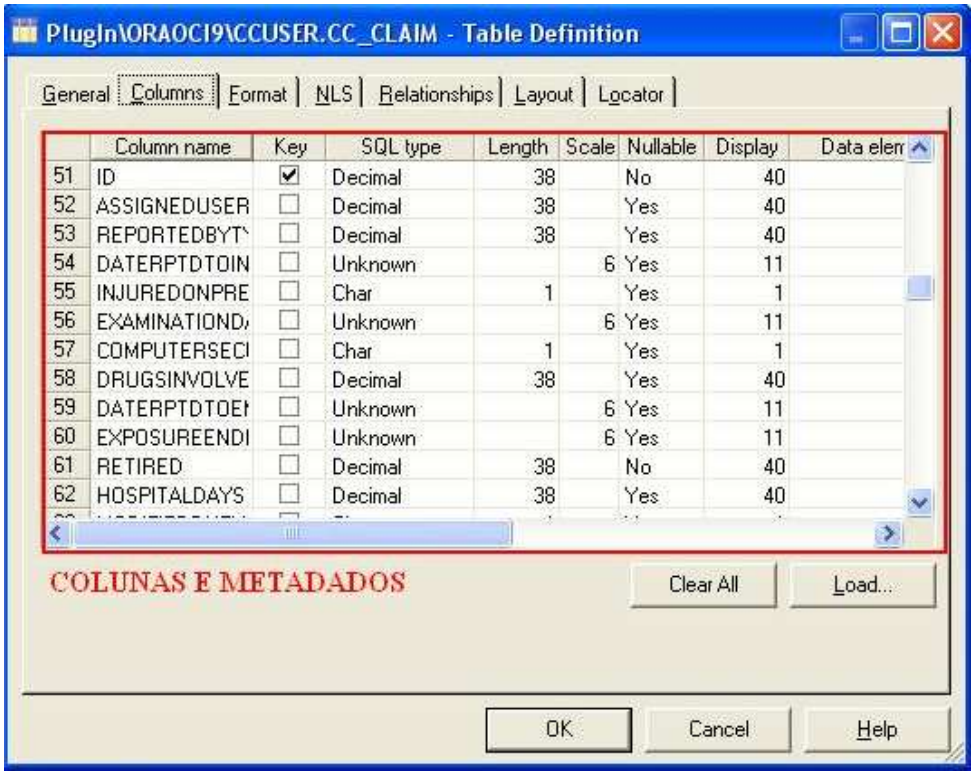


Figura 4.4 – Colunas e metadados

Com todas as tabelas importadas e seus metadados é dado o início do processo de geração dos arquivos.

4.2 O Processo de ETL – Geração do arquivo de datas

Esta seção mostra um processo ETL para geração de um arquivo para controle das datas de execução do projeto. Foi preciso desenvolver este *job* porque em reuniões com usuários, estes especificaram que a periodicidade da geração dos arquivos seria mensal, mas que não haveria uma data correta para isto. Com posse desta informação foi feita reunião com o DBA para ver ser era possível criar uma *flag* para controlar quais dados já foram tratados, mas nenhuma alteração neste sentido poderia ser feito. Então foi preciso criar uma lógica para atender este quesito. A figura 4.5 mostra o início deste *job*.

Figura 4.5 – Início do *job* geração do arquivo de datas

Este *job* tem origem em um estágio Oracle, onde é recuperada uma única linha utilizando o comando "ROWNUM <= 1". Este único registro é passado para o estágio *Transformer* CriaDataExecucao, e é recuperado a data atual do sistema. Esta data é armazenada no estágio *Sequential File* ArqDataExecucao. A cada execução é gravado ao final deste arquivo a data corrente. No estágio *Transformer* TrfContadorData é criado um contador que é incrementado a cada data diferente passada pelo estágio anterior. A segunda parte deste *job* é apresentada na figura 4.6.

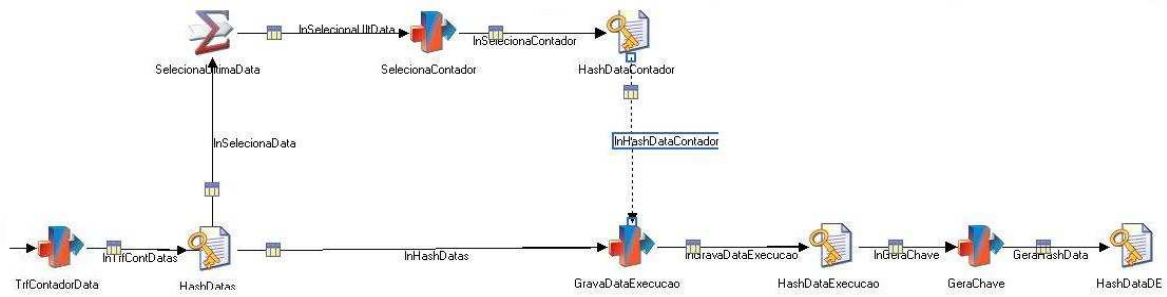


Figura 4.6 – Parte final do *job* geração do arquivo de datas

A figura 4.6 mostra os dados carregados no estágio *Hash* HashDatas, e então é dividido em dois fluxos: lnSelecionaData e lnHashDatas.

No fluxo lnSelecionaData os registros são passados para o estágio *Aggregator* SelecionaUltimaData onde é recuperada a última data com o seu respectivo contador. Os dados são passados para o estágio *Transformer* SelecionaContador onde é feito um tratamento para pegar o penúltimo contador, que é referente a data desejada de início. Este registro contador é gravado no estágio *Hash* HashDataContador, que é utilizado como *lookup* do fluxo lnHashDatas. Este *lookup*, mostrado na figura 4.7, é feito no estágio *Transformer* GravaDataExecucao.

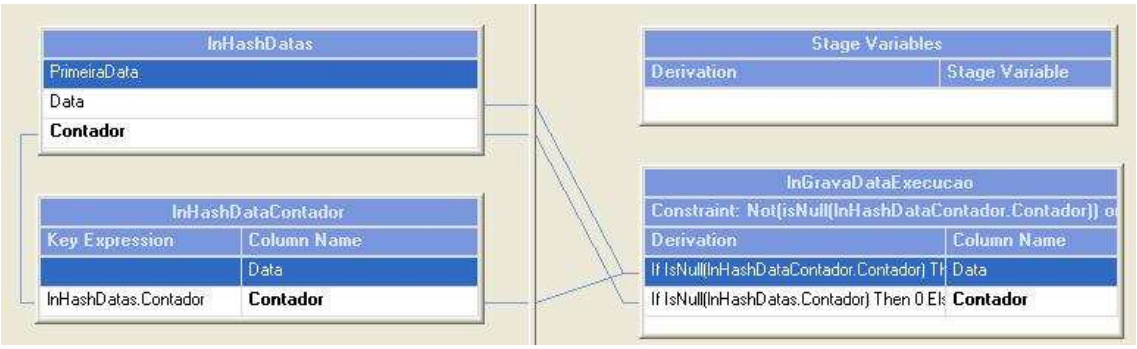


Figura 4.7 – *Lookup* GravaDataExecucao

O *lookup* é feito ligando o contador do fluxo lnHashDatas com o contador do fluxo lnHashDataContador. Com isto será passado para frente os dados onde a chave Contador, do fluxo de origem, bater com a do fluxo *lookup*. Porém na primeira execução estas chaves não irão bater pelo tratamento de decrementação do *lookup*. Para corrigir este problema foi feito um filtro em *Constraint* "Not(isNull(lnHashDataContador.Contador)) or lnHashDatas.Contador = 1". Este filtro significa que somente os dados em que a chave bater ou quando o contador do fluxo principal for igual a 1 irão seguir pelo fluxo. Quando lnHashDatas.Contador for igual a 1, uma data fictícia é utilizada para recuperar todos os registros. Da segunda execução em diante será recuperada sempre a penúltima data. Esta data é gravada no estágio *Hash* HashDataDE, para ser utilizado no *jobs* posteriores.

4.3 O Processo de ETL – Geração do arquivo de sinistros

O objetivo deste *job* é gerar um arquivo com todos os sinistros ocorridos em um período de aproximadamente um mês. Para o desenvolvimento deste *job* foi gerada a seguinte especificação:

Executar a *query* a seguir.

```
SELECT
        t1.underwritingco,
        t1.codunidadeemissora_Ext,
        t1.policynumber,
        t2.claimnumber,
        t2.lossDate,
        t3.codmarcatipo_Ext,
        t2.lossCause,
        t4.updateTime,
        t2.lossCause,
        t2.reportedDate,
        t1.nNumEndosso_Ext,
        e1.numdoefeito_ext,
        t5.datcomunicacao_Ext,
        t6.county,
        t6.city

    FROM
       ccuser.cc_policy                          t1,
```

```
    ccuser.cc_claim                          t2,
    ccuser.cc_vehicle                        t3,
    ccuser.cc_exposure                       t4,
    ccuser.cc_incident                       t5,
    ccuser.cc_address                        t6,
    ccuser.cctl_underwritingcompanytype      t7,
    ccuser.cctl_unidadeoperacional_Ext       t8,
    ccuser.cctl_losspartytype                t9,
    ccuser.ccx_efeito_ext                    e1,
    ccuser.cctl_losspartytype                e2
WHERE
    t1.UnderWritingCo            = t7.ID                 and
    t1.CodUnidadeEmissora_Ext    = t8.ID                 and
    t2.policyID                  = t1.ID                 and
    t3.ID                        = t5.VehicleID          and
    t5.claimID                   = t2.id                 and
    t4.claimID                   = t2.ID                 and
    t5.ClaimID                   = t2.ID                 and
    t6.id                        = t2.losslocationID     and
    t2.State                     = 2                     and
    t9.id                        = t4.lossparty          and
    t4.lossparty                 = e2.id                 and
    t9.l_pt_br                   = 'Segurado'            and
    e1.covsubtype_ext            = t4.coveragesubtype    and
    t4.retired                   = 0                     and
    t4.claimorder                = 1                     and
    t4.createtime                > data-parametro        and
    t4.createtime                < data-parametro        and
    Min(t4.createtime )
```

Efetuar a gravação do arquivo conforme leiaute abaixo, seguindo as observações.

- Os campos originais “numéricos” deverão ser completados com zeros a esquerda;
- Os campos originais “char” deverão ser completados com espaços a direita;
- O nome das colunas não devem aparecer no arquivo final;
- Os registros devem ser ordenados pela coluna NumSinistro;
- A primeira linha deverá conter 'ARQUIVO_SINISTROS';
- A última linha deverá conter 'TOTAL> XXXXXXXXX', onde XXXXXXXX é o total de registros gravados;
- As colunas devem estar separadas pelo delimitador '>'. Após a última coluna o delimitador é '<'.

Tabela	Coluna Origem	Coluna Destino	Formato	Regra
UnderWritingCompanyType	Type_Code	COD_CIA_EMIS	Char (4)	

unidade_operacional_ext	Type_Code	COD_UNID_EMIS	Char (4)	
cc_policy	PolicyNumber	NUM_APOLICE	Char (9)	
		NUM_ITEM_APOL	Char (9)	000000001
cc_claim	ClaimNumber	NUM_SINISTRO	Char (9)	
cc_claim	LossDate	DAT_SINISTRO	Char (10)	MM/DD/AAAA
		COD_UNID_REG	Char (4)	
cc_vehicle	CodMarcaTipo_Ext	COD_MARCA_TIPO	Char (9)	
		FLG_MARCA_TIPO	Char (1)	‘S’
		IDC_COSSEG	Char (1)	‘ ‘
		PCT_PART_CIA	Char (6)	'000.00'
cc_exposure	UpdateTime	UPDATE_TIME	Char (19)	MM/DD/AAAA
cc_claim	LossCause	COD_CAUSE	Char (4)	Pegar os dois últimos algarismos e somar com 50
		COD_ORIGEM_TRAN	Char (1)	‘S’
cc_claim	ReportedDate	COD_AVIS_SIN	Char (10)	MM/DD/AAAA
cc_policy	NumEndosso_Ext	NUM_ENDOSSO	Char (9)	
ccx_efeito_ext	numdoefeito_ext	COD_EFEITO	Char (4)	
cc_claim	reporteddate	DAT_INF_SIN	Char (10)	MM/DD/AAAA
cc_claim	LossDate	HORA_SINISTRO	Char (8)	
cc_address	County	NOME_BAIRRO	Char (20)	Completar com espaços a direita
cc_address	City	NOME_CIDADE	Char (30)	Completar com espaços a direita

O *job* desenvolvido é mostrado na figura 4.8.

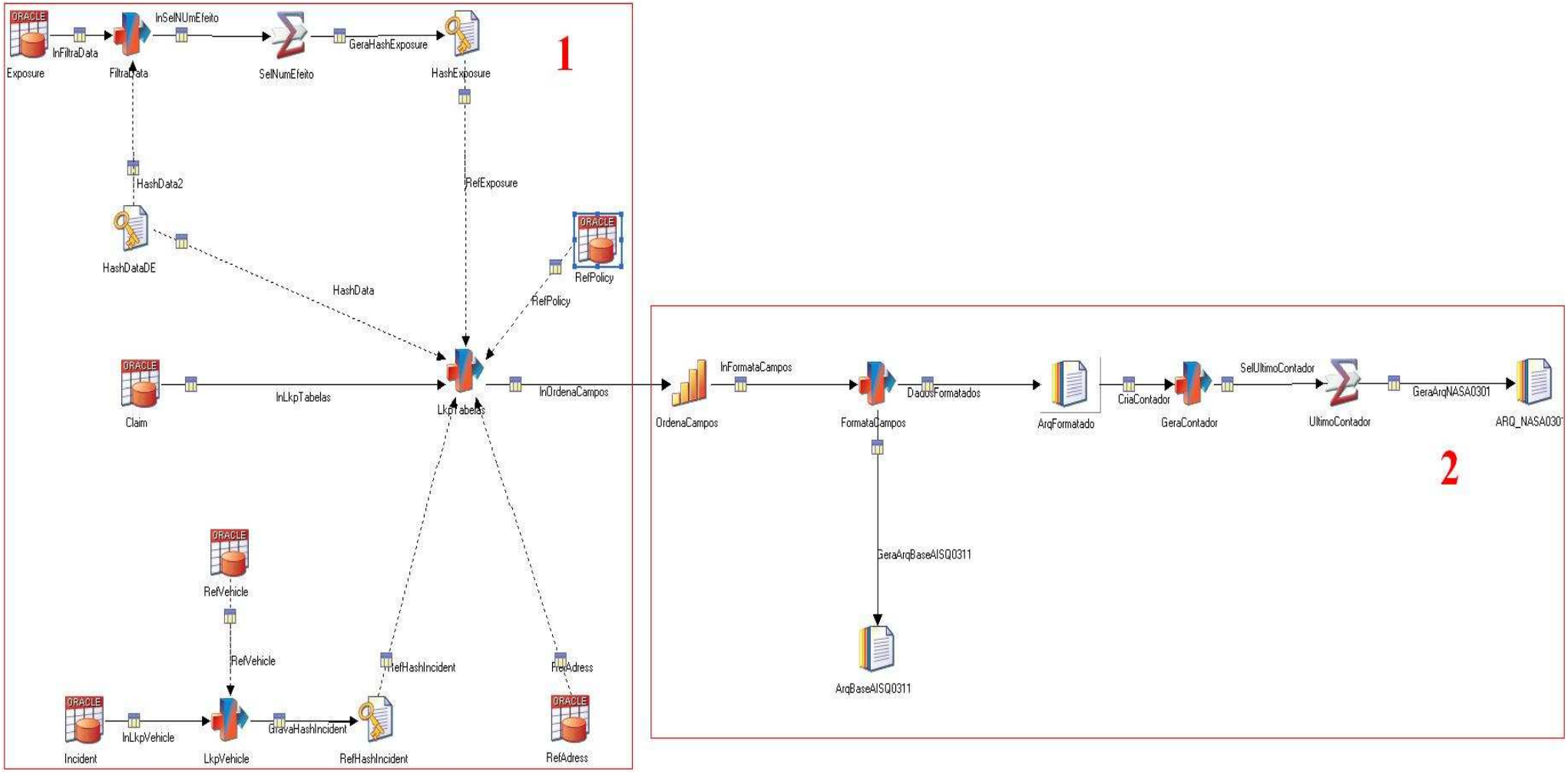
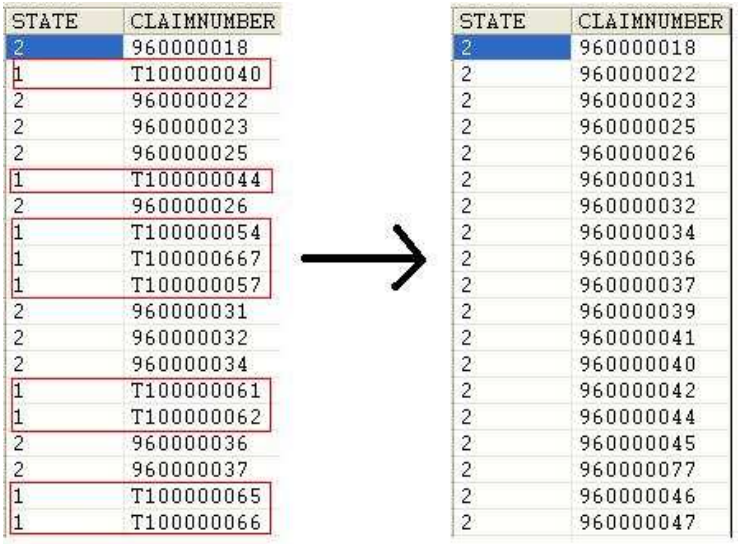


Figura 4.8 – Job Gera arquivos sinistros

A explicação do *job* responsável por gerar o arquivo de sinistros é dividida em duas partes. O início deste *job* é feito pelo estágio *Oracle* Claim. Neste estágio foram selecionadas as colunas ClaimNumber, LossDate, LossCause, ReportedDate necessárias para a geração do arquivo final e foi feito o seguinte filtro via query “CCUSER.CC_CLAIM.STATE = 2” para filtrar os registros temporários. A figura 4.9 exemplifica este filtro.



STATE	CLAIMNUMBER
2	960000018
1	T100000040
2	960000022
2	960000023
2	960000025
1	T100000044
2	960000026
1	T100000054
1	T100000667
1	T100000057
2	960000031
2	960000032
2	960000034
1	T100000061
1	T100000062
2	960000036
2	960000037
1	T100000065
1	T100000066

STATE	CLAIMNUMBER
2	960000018
2	960000022
2	960000023
2	960000025
2	960000026
2	960000031
2	960000032
2	960000034
2	960000036
2	960000037
2	960000039
2	960000041
2	960000040
2	960000042
2	960000044
2	960000045
2	960000077
2	960000046
2	960000047

Figura 4.9 – Filtro na tabela Claim

Com as tabelas Incident e Vehicle foi feito um *lookup*. Este *lookup* é feito no estágio Transformer LkpVehicle, ligando a coluna “VehicleId” da tabela Incident com a coluna “ID” da tabela Vehicle. Um filtro é feito dentro de *constraint* no *Transformer* para selecionar apenas os registros em que o *lookup* for verdadeiro. Após realizado o filtro, foram selecionadas as colunas CodMarcaTipo_Ext referente a COD_MARCA_TIPO e as colunas necessárias para realizar *lookup* com a tabela Claim. Apenas as colunas desejadas foram gravadas no estágio *Hash* RefHashIncident. A figura 4.10 mostra este passo.

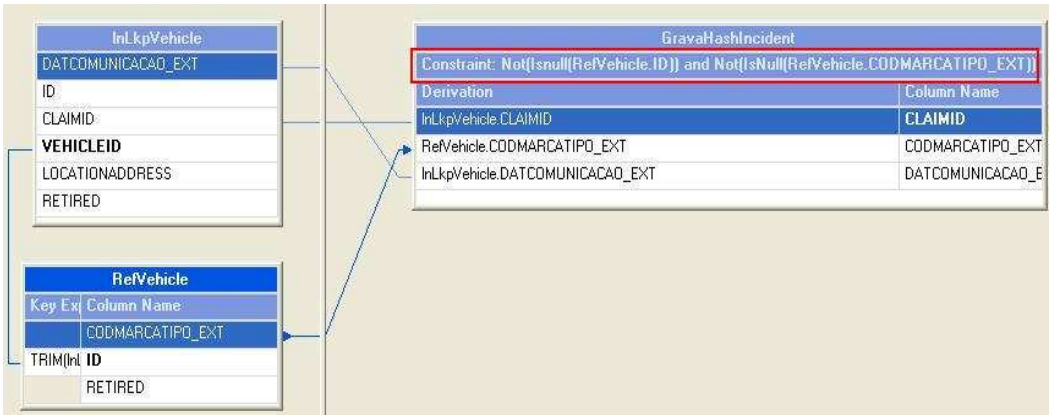


Figura 4.10 – *Lookup* Incident com Vehicle

No estágio *Oracle* RefPolicy é feito um join entre as tabelas Underwritingcomapanytype, Policy e Unidadeoperacional através da *query*:

“Underwritingcompanytype.ID = Policy.UNDERWRITINGCO and Unidadeoperacional_ext.ID = Policy.CODUNIDADEEMISSORA_EXT” . Da tabela UnderWritingCompanyType foi selecionada a coluna Type_Code responsável pelo preenchimento do campo COD_CIA_EMIS. Da tabela Unidade_operacional_ext foi recuperada a coluna Type_code referente a coluna COD_UNID_EMIS. E da tabela Policy foram recuperadas as colunas PolicyNumber e NumEndosso Referentes as colunas NUM_APOLICE e NUM_ENDOSSO, respectivamente.

Do estágio RefAdress foram selecionadas as colunas County e City referentes a NOME_BAIRRO e NOME_CIDADE.

No estágio *Oracle* Exposure foi feito um *join* entre as tabelas Exposure, Losspartytype, Efeito e Unidadeoperacional_ext através da *query*: “Losspartytype.ID = Exposure.LOSSPARTY and LosspartyType.L_PT_BR = 'Segurado' and Efeito_ext.COVSUBTYPE_EXT = Exposure.COVERAGESUBTYPE and Exposure.RETIRED = 0 and Exposure.CODUNIDADEREGULADORA_EXT = Unidadeoperacional_ext.ID and Expsoure.CLAIMORDER = 1”. Através deste *join* foi foram recuperadas as colunas Exposure.UpdateTime, Efeito_ext.Numdoefeito_ext e Unidadeoperacional_ext.Type_Code referentes a UPDATETIME, COD_EFEITO e COD_UNID_REGUL, respectivamente. Em seguida os dados foram repassados para o estágio *Transformer* FiltraData. Neste estágio foi feito um *lookup* com o arquivo Hash gerado no *job* anterior para realizar o filtro da data.

Para realizar este filtro foi preciso transformar a data contida na coluna CREATETIME da tabela Exposure, pois a mesma estava no padrão AAAA-MM-DD HH:MM:SSSSSS, para o formato AAAAMMDD. Para esta transformação foi utilizada a função *Field*: Field(lnFiltraData.CREATETIME,'-',1): Field(lnFiltraData.CREATETIME,'-',2):Field(Field(lnFiltraData.CREATETIME,' ' ,1), '-',3). Esta função seleciona uma parte do registro entre um delimitador. Por exemplo, em Field(lnFiltraData.CREATETIME,'-',1) é recuperado o ano, que é a primeira parte do campo CREATETIME.

Em seguida foi recuperado a data atual do sistema através da função Oconv(@DATE,"DYMD[4,2,2]"). Neste caso a data recuperada segue o formato AAAA MM DD. Os campos referente ao ano, mês e dia foram gravados em variáveis e concatenadas na variável DataAtual.

Em uma variável chamada FiltraData foi criado a condição do filtro de data: “If DataRegistro >= HashData2.Data and DataRegistro < DataAtual Then 1 Else 0”. Nesta

condição, a data referente ao CREATETIME é comparada com a data do arquivo Hash e com a data atual do sistema. Se a data CREATETIME estiver entre estas duas datas a variável FiltraData recebe o valor 1 senão recebe o valor 0. Então, um filtro é feito utilizando a variável FiltraData. A figura 4.11 mostra este tratamento de data e as colunas recuperadas do estágio *Oracle* Exposure.

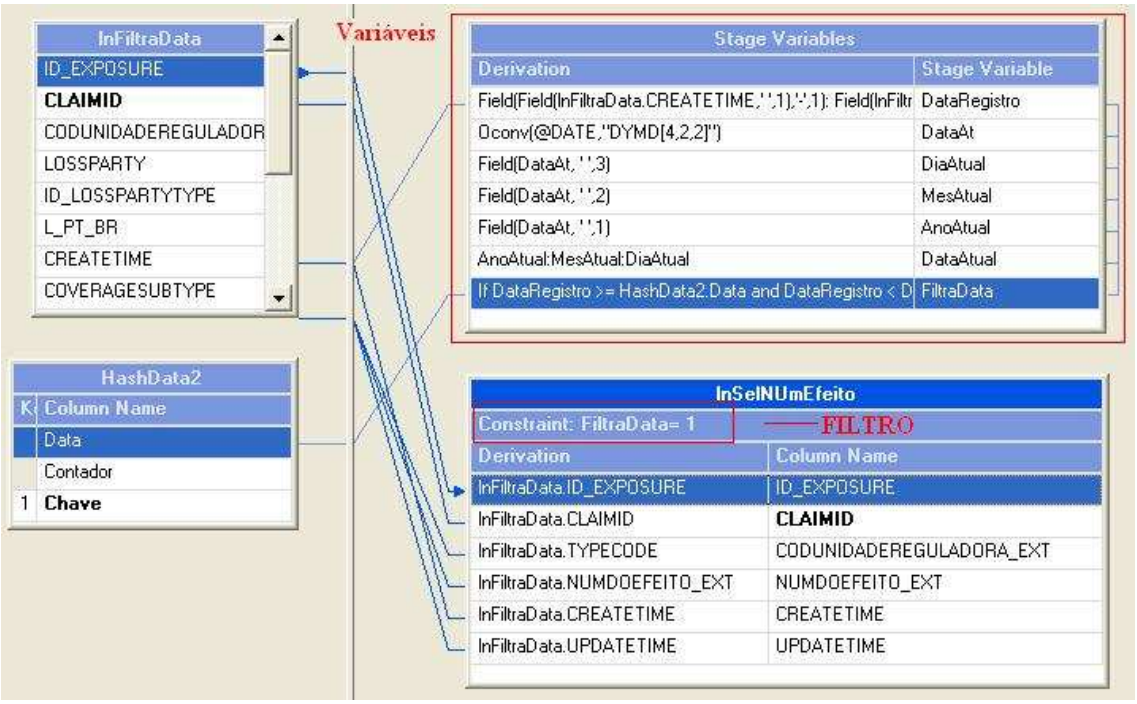


Figura 4.11 – Tratamento de data

Para recuperar o registro mínimo de Exposure.CREATETIME, conforme a *query* da especificação, os dados foram passados para o estágio *Aggregator* SelNumEfeito. Neste estágio os dados foram agrupados e a menor data referente a CREATETIME foi recuperada. Os dados foram, então gravados no estágio *Hash* HashExposure. O estágio *Aggregator* é exemplificado na figura 4.12.

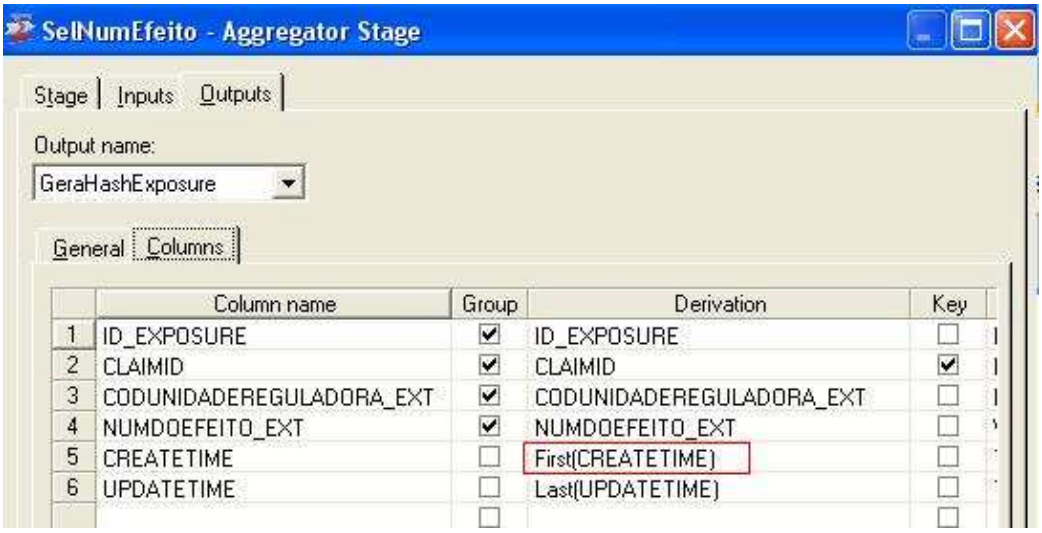


Figura 4.12 – Estágio Aggregator Data Mínima

No estágio *Transformer* LkpTabelas é feito a ligação da tabela Claim com os arquivos *Hashs* RefHashIncident e HashExposure, e com as tabelas Policy e Adress. Ainda neste estágio é feito algumas transformações nos dados e um filtro. Há transformações referente a data que utilizam a função *Field*, demonstrada anteriormente: Field(lnLkpTabelas.LOSSDATE, '-', 2):'/': Field(Field (lnLkpTabelas.LOSSDATE, '-',3), ' ',1) :'/':Field(lnLkpTabelas.LOSSDATE,'-',1). Neste caso a data possui o delimitador '/'. Ainda neste estágio é feito um tratamento para os registros referente a COD_CAUSE: "Substrings(lnLkpTabelas.LOSSCAUSE,4,2) + 50". Neste tratamento foi utilizada a função *Substrings*, e foi recuperado os dois últimos dígitos dos registros LOSSCAUSE e acrescentado 50 a estes registros. A figura 4.13 mostra este estágio.

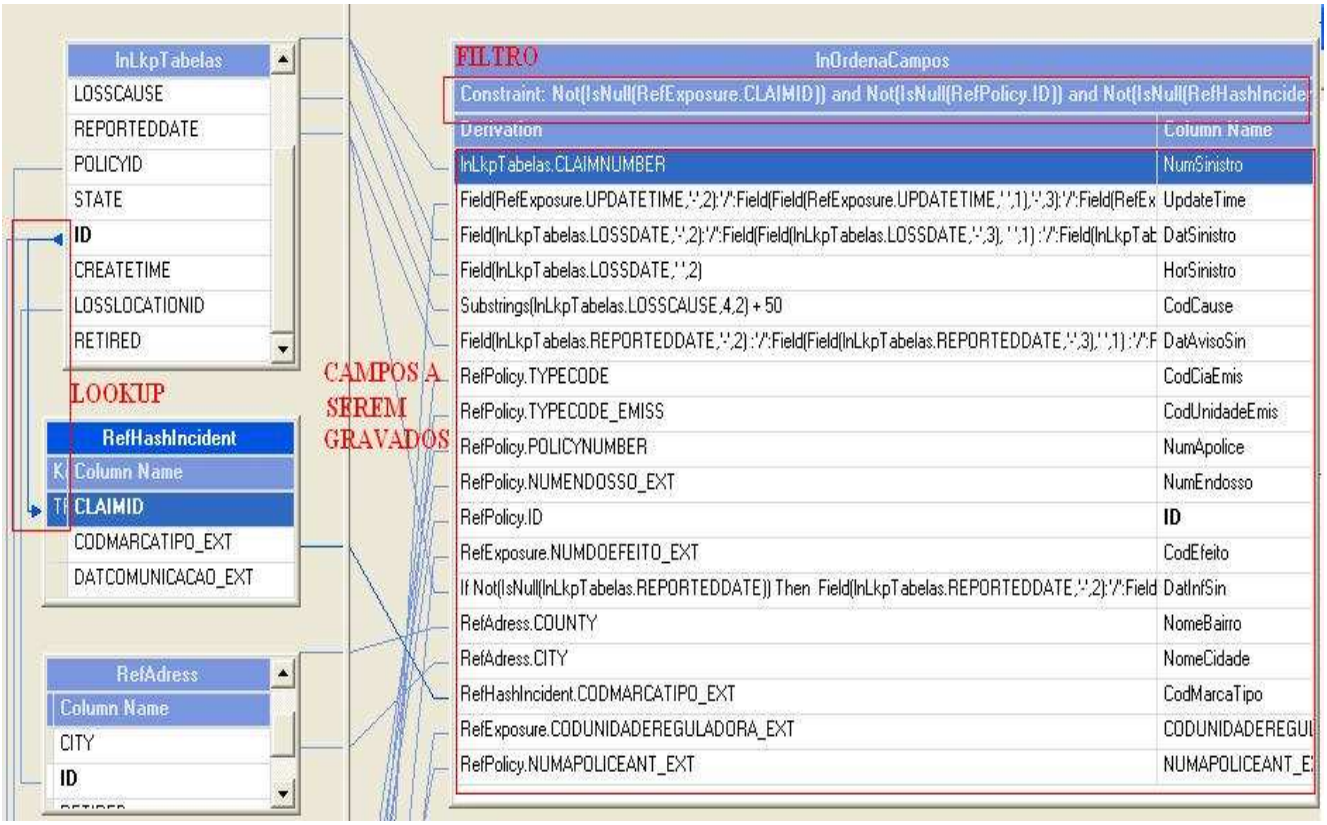


Figura 4.13 – Estágio *Transformer* LkpTabelas

A segunda parte deste job, conforme a figura 4.8, inicia-se no estágio *Sort* OrdenaCampos que recebe os dados advindos do estágio *Transformer* LkpTabelas e realiza uma ordenação através do campo NumSinistro. A figura 4.14 exemplifica esta ordenação.

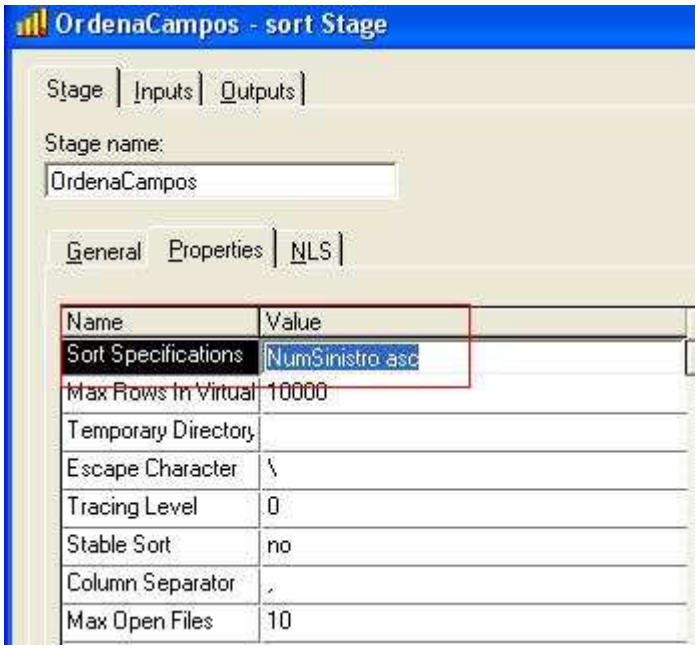


Figura 4.14 – Estágio Sort OrdenaCampos

Após a ordenação dos campos, os dados foram passados para o estágio *Transformer* FormataCampos onde foi feito o tratamento de nulidade para todos os campos. Este tratamento é feito pelo seguinte comando: “If IsNull(lnFormataCampos.CodCiaEmis) then '-' Else Str(0,(4 -Len(lnFormataCampos.CodCiaEmis))):lnFormataCampos.CodCiaEmis”. Neste caso se o campo for nulo é preenchido com um ‘-‘ e completado com espaços à direita. Se o campo não for nulo, é utilizada a função “Str” para completar com zeros à esquerda. Depois de feito este tratamento os registros foram gravados em uma coluna com o nome do arquivo “SINISTRO”. Com isto tem o nome do arquivo como cabeçalho e os registros gravados no arquivo. Em outro arquivo as colunas foram gravadas para ser utilizada em um processo posterior. O tratamento dos campos e a geração dos arquivos são apresentados na figura 4.15.

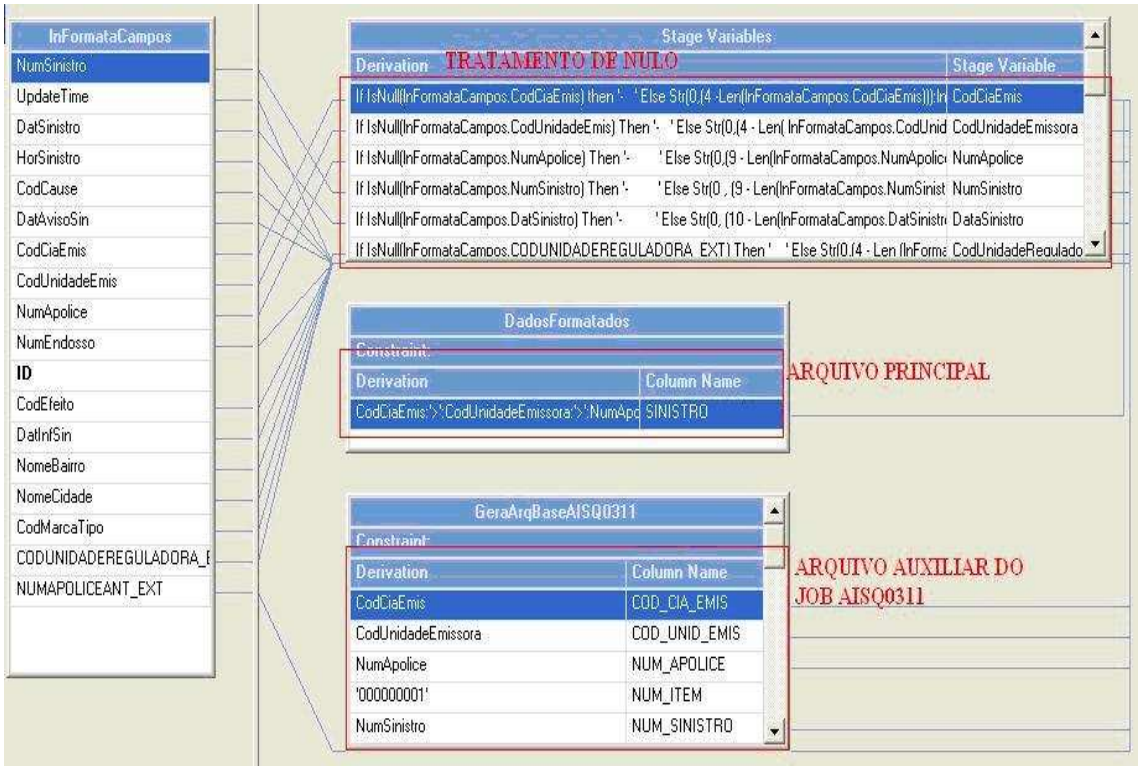


Figura 4.15 – Estágio *Transformer* FormataCampos

O próximo passo é gerar o rodapé com o número total de linhas. Para isto os registros foram passados para o estágio *Transformer* GeraContador. Neste estágio foi criada uma variável “Contador” e a cada registro passado esta variável é incrementada. Para realizar a gravação do rodapé foi criada uma coluna com a seguinte derivação: “'TOTAL>':' ':Str(0, (9 - Len(Contador))):Contador”. Nesta derivação é gravado o total de linhas precedido de “TOTAL>”. A figura 4.16 apresenta este passo.

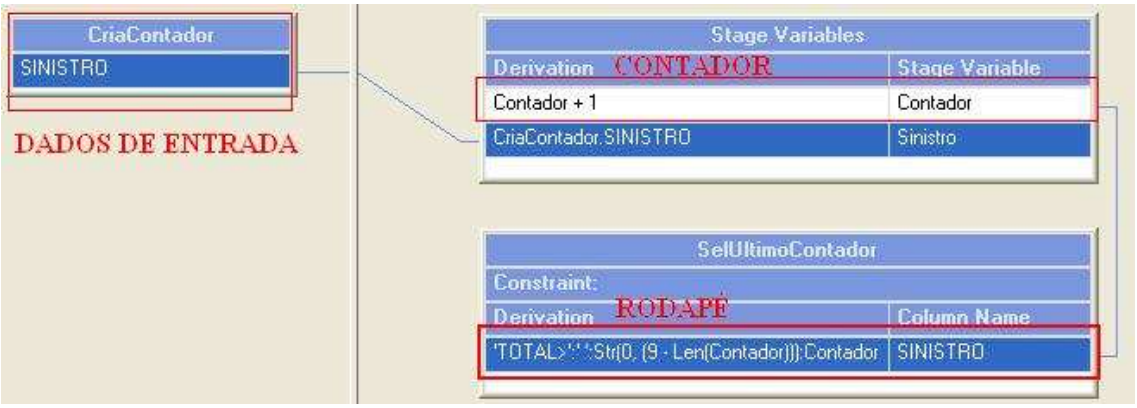


Figura 4.16 – Estágio *Transformer* GeraContador

O próximo passo é selecionar o último registro, referente ao número total de linhas lidas. Para isto foi utilizado o estágio *Aggregator* UltimoContador. Neste estágio foi utilizada a função “Last” para retornar o último contador. A figura 4.17 apresenta esta etapa.

Figura 4.17 – Estágio *Aggregator* UltimoContador

Por fim, este contador foi inserido ao final do arquivo principal mostrado na Figura 4.15. Para isto, foi utilizado o estágio *Sequential File* com a opção “*Append to existing file*” para inserir o contador no final do arquivo. A figura 4.18 apresenta esta opção.

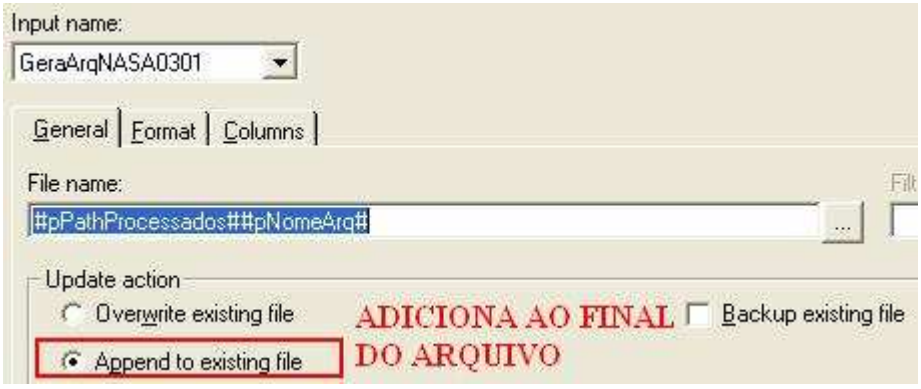


Figura 4.18 – Rodapé do arquivo Sinistros

Após realizar todos estes passos o arquivo gerado é mostrado na figura 4.19.

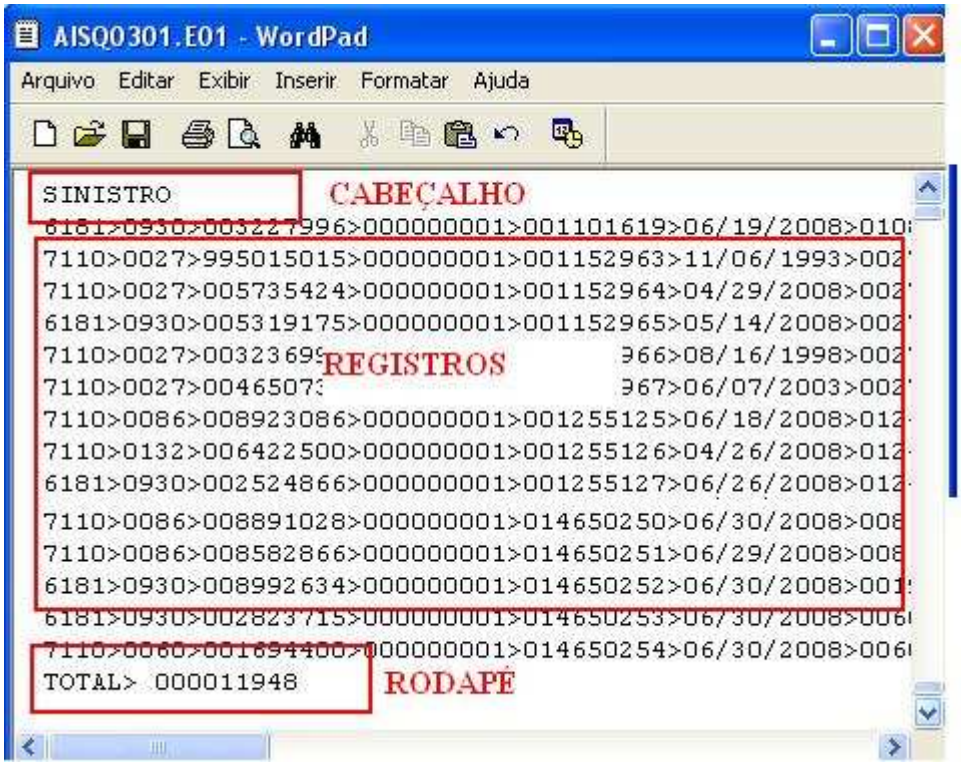


Figura 4.19 – Resultado dos arquivos Sinistro

4.4 O Processo de ETL – Geração do arquivo histórico sinistros

A geração do arquivo histórico de sinistros é feito pelo *job* GeraArqHistoricoSinistros. Para o desenvolvimento deste job foi gerada a seguinte especificação:

Especificação:

Para cada linha recuperada do *job* AISQ0301, realizar um *join*, utilizando o número do sinistro, com a tabela historico_sinistro_ext e gravar todas as linhas onde historico_sinistro.tipobeneficio e historico_sinistro.autorepair for diferente de nulo.

Desenvolver conforme a *query* abaixo.

```
SELECT
      t3.username
FROM
     ccuser.ccx_historico_sinistro_ext    t1,
     ccuser.cc_user                       t2,
     ccuser.cc_credential                 t3

WHERE
      ArquivoBaseAISQ0311.numsinistro = t1.numsinistro_ext and
     t1.updateuserid                  = t2.id              and
    t2.credentialid                   = t3.id
```

Efetuar a gravação do arquivo conforme leiaute a seguir.

- O nome do arquivo a ser gerado é AISQ0311;
- Os campos originais “numéricos” deverão ser completados com zeros a esquerda;
- Os campos originais “char” deverão ser completados com espaços a direita;
- O nome das colunas não devem aparecer no arquivo final;
- A primeira linha deverá conter 'HIST_OPCAO_DPI_VIPRA';
- A última linha deverá conter 'TOTAL> XXXXXXXXX', onde XXXXXXXX é o total de registros gravados;
- As colunas devem estar separadas pelo delimitador '>'. Após a última coluna o delimitador é '<'.

Tabela	Coluna Origem	Coluna Destino	Formato	Regra
UnderWritingCompanyType	Type_Code	COD_CIA_EMIS	Char (4)	
unidade_operacional_ext	Type_Code	COD_UNID_EMIS	Char (4)	
policy	PolicyNumber	NUM_APOLICE	Char (9)	
		NUM_ITEM_APOL	Char (9)	000000001
claim	ClaimNumber	NUM_SINISTRO	Char (9)	
historico_Sinistro _Ext	TipoBenefcicio_Ext	IDC_OPCAO_DPI	Char (2)	Se Tipobeneficio_ext = 10001 então 'CR' Senão 'DF'
historico_Sinistro _Ext	UpdateTime	DAT_ULT_ALT	Char (19)	
credential	UserName	SIG_USUARIO_ALT	Char (8)	
historico_Sinistro _Ext	‘FALTA DEFINIR’	SIG_MODULO	Char (8)	

O *job* desenvolvido é mostrado na figura 4.20.

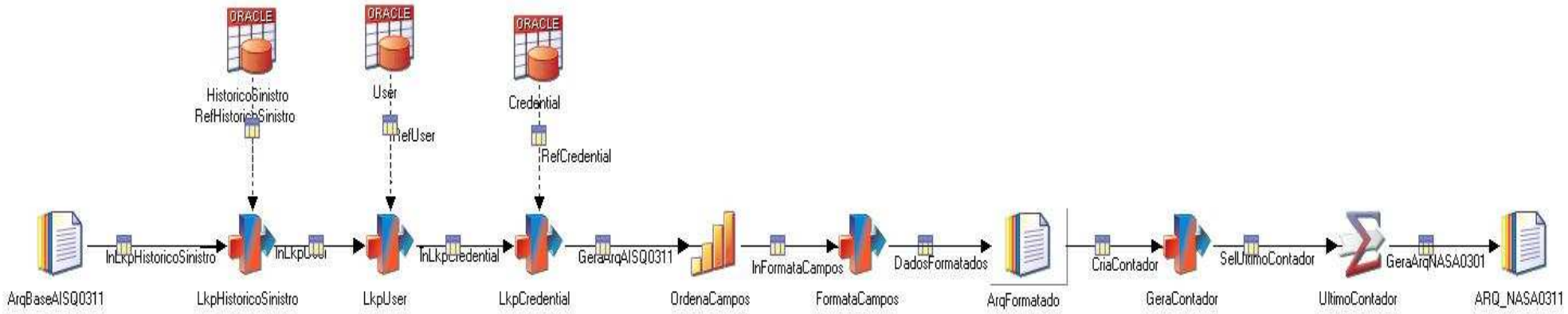


Figura 4.20 – *Job* Geração do arquivo Histórico de Sinistros

Este *job* tem como arquivo de origem o arquivo base gerado no job anterior. As colunas deste arquivo são apresentadas na Figura 4.21.

Stage | Outputs

Output name: InLkpHistoricoSinistro — View Data...

General | Format | Columns — COLUNAS DO ARQUIVO BASE

	Column name	Key	SQL type	Length	Scale	Nullable	Display	Dat
1	COD_CIA_EMIS		Char	4		No		
2	COD_UNID_EMIS		Char	4		No		
3	NUM_APOLICE		Char	9		No		
4	NUM_ITEM		Char	9		No		
5	NUM_SINISTRO		Char	9		No		
6	DATA_SINISTRO		Char	10		No		
7	COD_UNIDADE_REGUL		Char	4		No		
8	COD_MARCA_TIPO		Char	9		No		
9	FLG_MARCA_TIPO		Char	1		No		
10	IDC_COSSEG		Char	1		No		
11	PCT_PART_CIA		Char	6		No		
12	UPDATETIME		Char	19		No		
13	COD_CAUSE		Char	4		No		
14	COD_ORIGEM_TRAN		Char	1		No		
15	COD_AVISO_SIN		Char	10		No		
16	NUM_ENDOSSO		Char	9		No		
17	COD_EFEITO		Char	4		No		
18	DAT_INF_SIN		Char	10		No		
19	HORA_SIN		Char	8		No		
20	NOME_BAIRRO		Char	20		No		
21	NOME_CIDADE		Char	30		No		

Figura 4.21 – Colunas do arquivo base

A partir do arquivo de origem foram selecionadas as colunas desejadas e feito um *lookup* Historico_Sinistro, através do estágio *Transformer* LkpHistoricoSinistro. Após selecionado as colunas foi feito um filtro para selecionar as linhas onde historico_sinistro.tipobeneficio e historico_sinistro.autorepair fossem diferente de nulo. Com isto o objetivo inicial da especificação foi atingido. Ainda neste estágio foi feito uma trasnformação nos dados da seguinte forma “If RefHistoricoSinistro.TIPOBENEFICIO_EXT = 10001 Then 'CR' Else 'DF'”. Com esta transformação, os dados referentes a TIPOBENEFICIO_EXT que forem iguais a 10001 são gravados com ‘CR’, caso contrário é gravado ‘DF’. Este estágio é apresentado na figura 4.22.

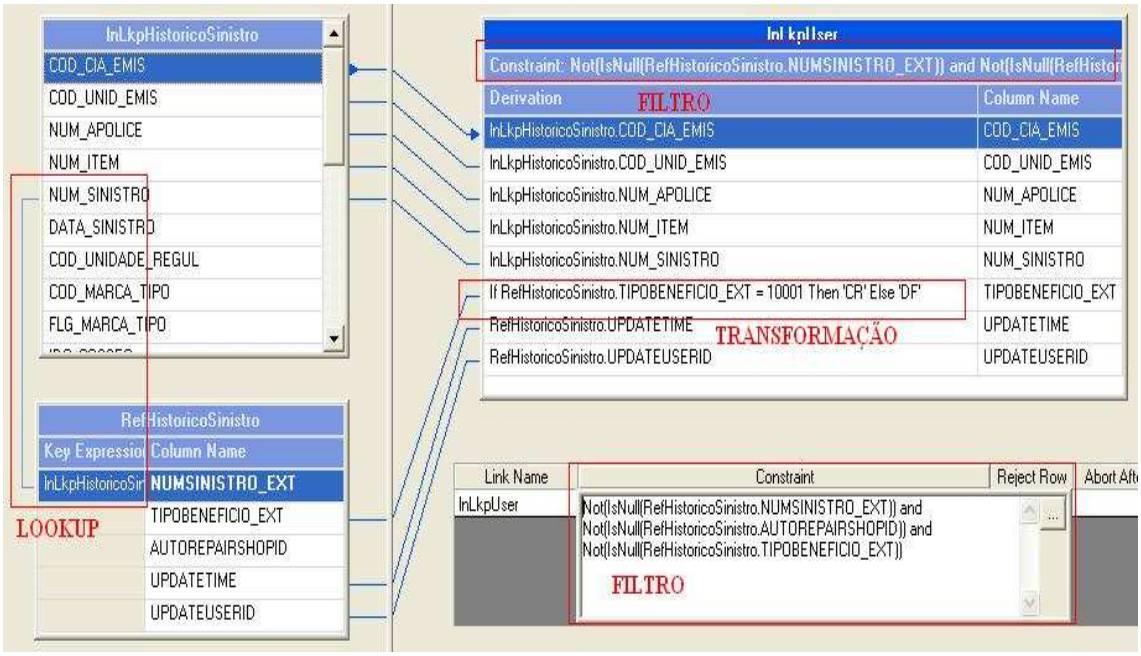


Figura 4.22 – Estágio *Transfomer* LkpHistoricoSinistro

Em seguida é feito um *lookup*, no *Transformer* LkpUser, com a tabela User para recuperar a coluna CREDENTIALID para ser utilizado no *lookup* com a próxima tabela. O *lookup* feito no *Trasnformer* LkpUser é apresentado na figura 4.23.

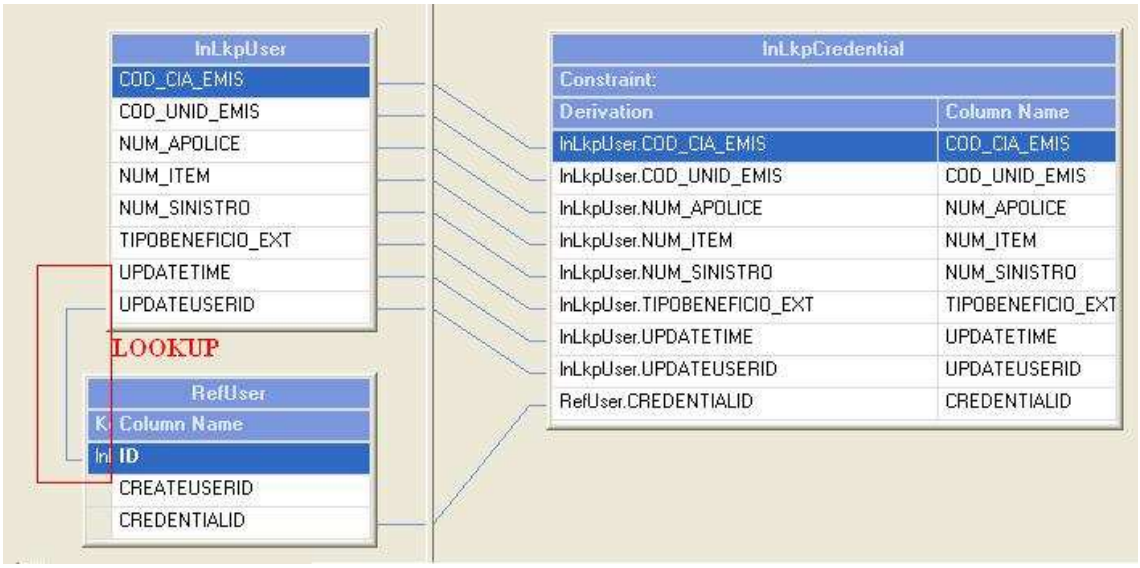


Figura 4.23 – *Transformer* LkpUser

No *Transformer* LkpCredential é feito o *lookup* com a tabela Credential e é feito o tratamento para a data UPDATETIME da mesma forma feita nos processos anteriores. Esta transformação é da forma “Field (Field (lnLkpCredential.UPDATETIME, ' ', 1), '-', 3):'/': Field(lnLkpCredential.UPDATETIME, '-', 2):'/': Field(lnLkpCredential.UPDATETIME, '-', 1):' ':Field(lnLkpCredential.UPDATETIME, ' ', 2)”. Com isto transformação a data é gravada no formato DD/MM/AAAA HH:MM. É feito, também, uma padronização para a coluna

SIG_USUARIO, utilizando a função “Str”: “RefCredential.USERNAME:Str(' ', (8 - Len(RefCredential.USERNAME)))”. Esta padronização acrescenta espaços a direita dos dados USERNAME. Este estágio é mostrado na figura 4.24.

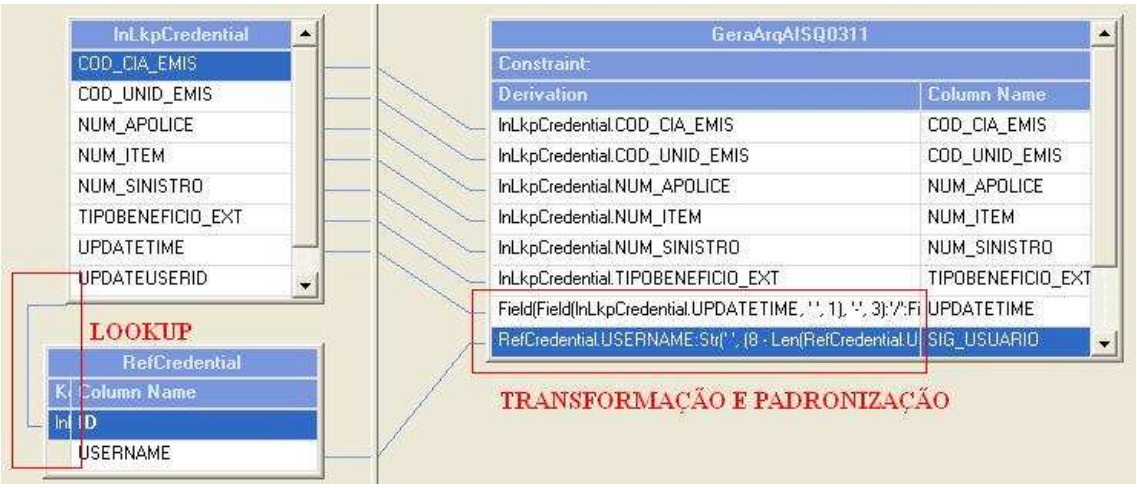


Figura 4.24 – Estágio LkpCredential

Os dados selecionados foram passados para o estágio *Sort* OrdenaCampos. Neste estágio os dados foram ordenados por NUM_APOLICE, NUM_SINISTRO. Este estágio é apresentado a Figura 4.25.

Figura 4.25 – Estágio OrdenaCampos

Com os dados ordenados, estes foram passados para o estágio *Tranformer* FormataCampos. Neste estágio foi feito uma transformação passando para letras maiúsculas os dados de SIG_USUARIO e foram gravados os dados em uma coluna com o nome do arquivo. Com isto o cabeçalho e os dados padronizados foram gravados no estágio *Sequential File* ArqFormatado. A figura 4.26 apresenta este estágio.

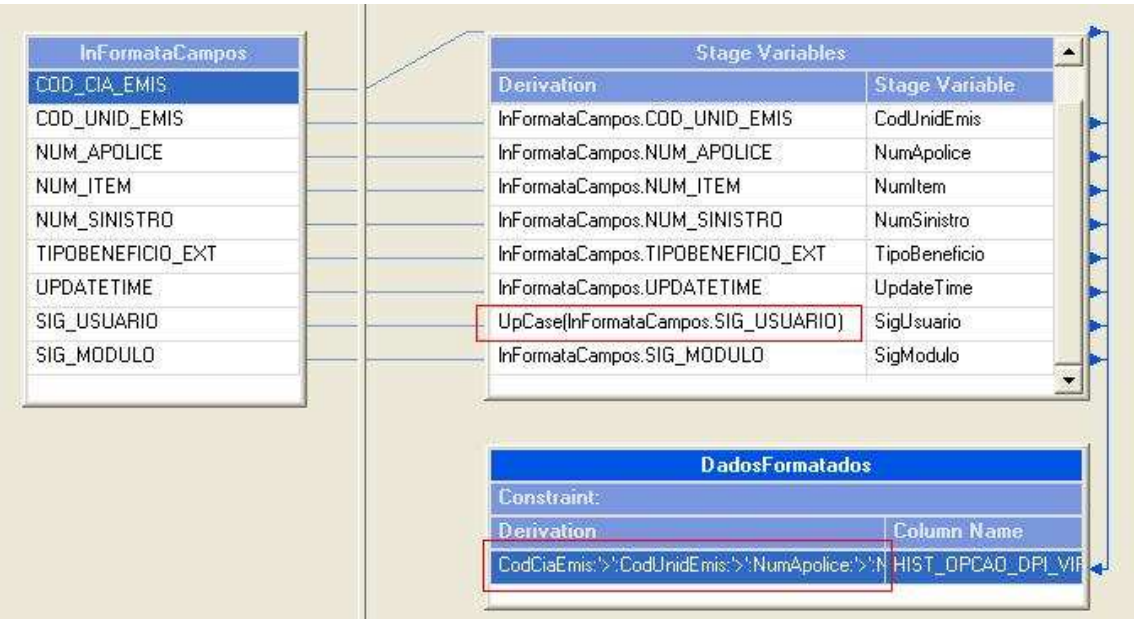


Figura 4.26 – Estágio *Transformer* PadronizaCampos

O próximo passo foi gerar um contador para retornar o total de registros. Este passo foi feito da mesma forma explicada no processo anterior, gerando um contador no estágio *Tranformer* GeraContador, que incrementa a cada linha lida, e em seguida no estágio *Aggregator* UlimoContador foi selecionado o último contador para retornar o número de linhas lidas, através da função "*Last*". Este total de linhas lidas é acrescentada ao final do arquivo, compondo o rodapé do arquivo. Os dados foram, então, gravados em um estágio *Sequential_File*. O resultado deste arquivo pode ser conferido na figura 4.27.

```
HIST_OPCAO_DPI_VIPRA
7110>0027>005735424>000000001>001152964>FC>06/19/2008 09:13:40>FD005463>APA4380 <
6181>0930>005319175>000000001>001152965>FC>06/19/2008 10:18:05>FD005463>APA4380 <
6181>0930>002524866>000000001>001255127>FC>07/02/2008 11:01:08>FD019321>APA4380 <
7110>0086>008922624>000000001>001612559>FC>06/24/2008 09:52:39>DL071779>APA4380 <
7110>0027>005896002>000000001>007387089>FC>06/19/2008 16:21:38>MJ013267>APA4709 <
6181>0930>005477525>000000001>007596383>FC>06/25/2008 10:17:08>RS337328>APA4709 <
6181>0930>005336477>000000                                  :57:19>FC343921>APA4709 <
6181>0930>005475485>000000                                  :59:20>AP017731>APA4709 <
7110>0060>001117690>000000                                  :01:10>FL021749>APA4709 <
7110>0132>006458580>000000001>008020791>FC>06/23/2008 11:36:26>AP017731>APA4709 <
7110>0060>008472607>000000001>008027936>DF>06/20/2008 14:46:03>MO002100>APA4709 <
6181>0930>002907735>000000001>014650247>FC>07/01/2008 01:14:37>APASF13 >APASF13 <
7110>0060>009925320>000000001>014650248>FC>07/01/2008 01:14:37>APASF13 >APASF13 <
6181>0930>002571565>000000001>014650249>FC>07/01/2008 01:14:37>APASF13 >APASF13 <
7110>0086>008891028>000000001>014650250>FC>07/01/2008 01:14:37>APASF13 >APASF13 <
7110>0086>008582866>000000001>014650251>FC>07/01/2008 01:14:38>APASF13 >APASF13 <
6181>0930>008992634>000000001>014650252>FC>07/01/2008 01:14:38>APASF13 >APASF13 <
6181>0930>002823715>000000001>014650253>FC>07/01/2008 01:14:38>APASF13 >APASF13 <
7110>0060>001694400>000000001>014650254>FC>07/01/2008 01:14:38>APASF13 >APASF13 <
TOTAL> 000012827
```

CABEÇALHO

AMOSTRAGEM DOS DADOS SELECIONADOS

RODAPÉ

Figura 4.27 – Arquivo Histórico Sinistro

4.5 O Processo de ETL – Sequenciamento dos Jobs

Após a construção de todos os *jobs*, é preciso que estes executem em uma determinada ordem. Para realizar esta tarefa foi criado um *job Sequence* (figura 4.28).

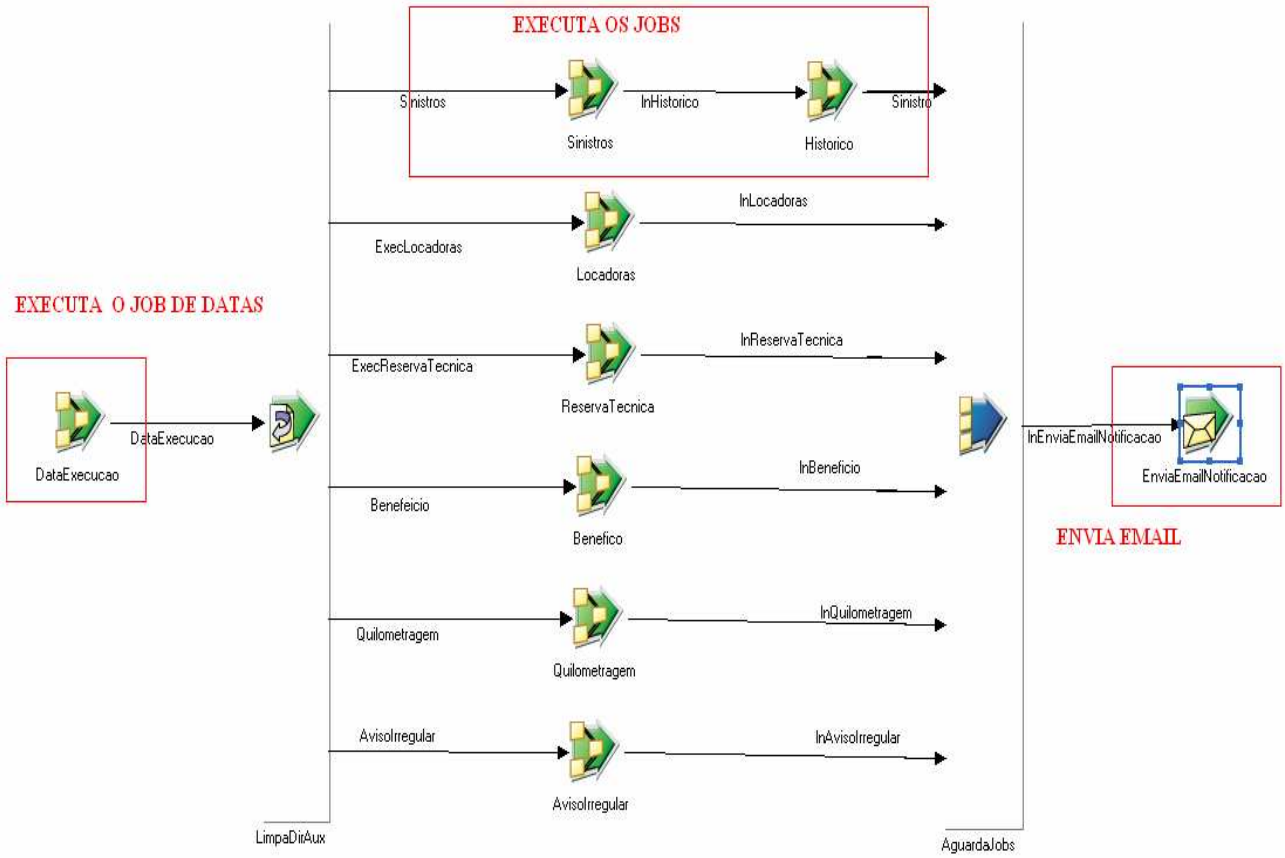


Figura 4.28 – *Job* Sequence

Este *job* inicia-se com a execução do *job* responsável por gerar o arquivo de datas, demonstrado na seção 4.2. Em seguida é utilizado o estágio *Execut Command* LimpaDirAux para executar a limpeza do diretório de arquivos auxiliares toda vez que o fluxo for executado. Para executar esta limpeza foi utilizado o comando "ls" e selecionado o parâmetro dos diretórios temporários. Este comando é apresentado na figura 4.29.

Figura 4.29 – Limpeza do diretório

Após a limpeza do diretório, passou-se para a execução dos *jobs* responsáveis pela geração dos arquivos. Em destaque na figura 4.28 estão os dois *jobs* demonstrados nas seções

4.3 e 4.4. O *job* responsável por gerar o arquivo Histórico_Sinistro só inicia após a completa execução do *job* responsável pela geração do arquivos de sinistros. Simultaneamente a execução do *job* ArquivoSinistro, são executados os *jobs* responsáveis pela geração dos outros arquivos. O estágio *Sequencer* AguardaJobs é utilizado para que o próximo passo só inicie após a finalização de todos os outros estágios. Após todos *jobs* serem executados, um email de notificação é enviado aos responsáveis pelo processo informando se o processo ocorreu com sucesso ou se ocorreu alguma falha. Ainda neste email é acrescentado um *log* de todos os *jobs* informando as características de cada estágio. Esta notificação é feita no estágio *Notification Activity* EnviaEmailNotificacao, apresentado na figura 4.30.

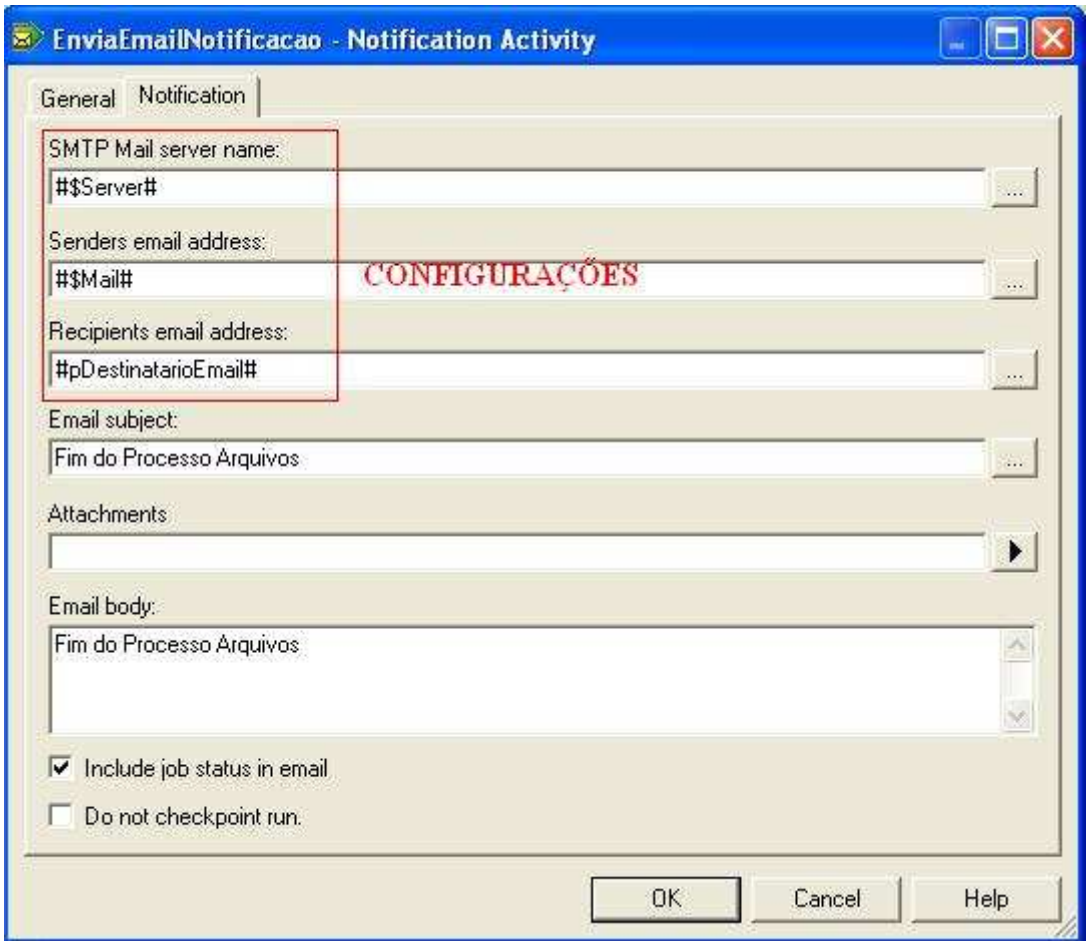


Figura 4.30 – Notificação por email

A figura 4.30 mostra as configurações necessárias para enviar o email de notificação do final do processo de geração dos arquivos. Em "*SMTP Mail Server name*" é colocado o nome do servidor SMTP. Em "*Senders email adress*" é colocado o email do remetente. Em "*Recipients email adress*" é colocado o email dos destinatários responsáveis pelo processo.

Depois de finalizado este *job*, os arquivos gerados são enviados para os usuários finais que os analisam para verificar se há alguma inconsistência ou irregularidade com os

dados. Se constatado algum problema é preciso alterar o *job* responsável pelo arquivo analisado, caso contrário os *jobs* são passados para a etapa de produção.

CAPÍTULO 5
CONCLUSÃO E CONSIDERAÇÕES FINAIS

O surgimento dos SGBDs permitiu que instituições e empresas armazenassem grandes volumes de dados sobre suas operações. Além deste grande volume de dados, a inconsistência e descentralização dos bancos fazem com que realizar uma análise manual se torne impraticável. Visando facilitar a interpretação dos dados surgiram o DW e processo de KDD. O primeiro é um banco de dados especial utilizado para armazenar dados relativos às atividades de uma organização, de forma consolidada. O segundo tem objetivo de gerar conhecimento a partir de uma base de dados. Para que estes processos sejam realizados com sucesso é preciso que seja feito um tratamento dos dados. Esta etapa de tratamento dos dados é conhecida como ETL.

Este trabalho apresentou as principais características e conceitos do DW, do KDD e, de uma forma mais aprofundada, do ETL. Em seguida foram apresentadas as principais ferramentas de ETL do mercado e escolhida a ferramenta *DataStage* para realizar o estudo de caso. O estudo de caso apresentado é de seguradora e é um projeto especificamente de ETL, sem o objetivo de utilizar técnicas de mineração de dados ou construir um DW.

Através deste trabalho foram obtidos os seguintes ganhos:

- Livros e artigos foram lidos para revisar os conceitos de DW, KDD e ETL.
- Além da etapa de ETL que envolvem o DW e o processo de KDD
- Algumas ferramentas de ETL foram analisadas e foi feito um estudo mais detalhado em cima da ferramenta *DataStage*.
- Foi visto de forma prática como é feito o processo ETL envolvendo um grande banco de dados.

As principais dificuldades encontradas para realizar este trabalho foram:

- Encontrar livros que explorem com mais detalhes o processo ETL.
- Obter uma base de dados que servisse para realizar o processo prático.
- Muito tempo foi gasto para o estudo de DW, KDD e ETL.

Como sugestão para projetos futuros:

- Realizar um estudo mais aprofundado de uma ferramenta ETL livre.
- Construir um DW, utilizando a ferramenta ETL apresentada neste trabalho.

- Realizar, de forma prática, um estudo sobre mineração de dados com o auxílio de uma ferramenta própria para isto.

REFERÊNCIAS BIBLIOGRÁFICAS

BARBIERI, C.. BI – *Business Intelligence* – Modelagem & Tecnologia. Rio de Janeiro: Editora Axcel Books. 2001. 418p.

BOSCARIOLI, C.. Pré-processamento de Dados para a Descoberta de Conhecimento em Banco de Dados: Uma Visão Geral. Paraná. 2002. Universidade Estadual do Oeste do Paraná.

CARVALHO, D. R.; BUENO, M.; NETO, W. A.; LOPES,. L. R. Ferramenta de Pré e Pós-processamento para *Data Mining.* Paraná. 2002. 9p. Universidade Tuiuti do Paraná.

CHAUDHURI, S.; **DAYAL, U.** *An Overview of Data Warehousing and OLAP Technology.* 1997.

FAYYAD, U.; SHAPIRO, P. G.; SMYTH, P. ***Knowledge Discovery and TAData Mining: Towards a Unifying Framework***. 1996a.

FAYYAD, U.; SHAPIRO, P. G.; SMYTH, P. *The KDD Process for Extracting Useful Knowledge from Volumes of Data.* 1996b. 34p.

FARO, F.. Banco de Dados. 2001. Universidade de Brasília. Brasília.

GONÇALVES, M.. **Extração de dados para *Data Warehouse***. Rio de Janeiro – RJ. Axcell Books, 2003. 149p.

INMON, W. H.; WELCH, J.D.; GLASSEY, Katherine L. Gerenciando *DataWarehouses*. São Paulo: Editora Makron Books, 1999.375p.

KIMBALL, R.; CASERTA, J.. ***The Data Warehouse ETL Toolkit – Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data***. Indianápolis. Wiley Publishing, Inc., 2004. 490p.

KIMBALL, R.; ROSS, M.. ***The Data Warehouse Toolkit – The Complete Guide to Dimension Modeling***. Toronto. Wiley Publishing, Inc., 2002. 417p.

MARTINS, T. O.. *Data Warehouse:* Fundamentos, Tecnologia e Mecanismos de Análise. Juiz de Fora – MG. 2003. Universidade Federal de Juiz de Fora.

MELO, R.. Curso de *Data Warehouse.* 2000.

OLIVEIRA, W. J.. ***Data Warehouse***. Florianópolis – SC. Visual Books, 2002. 187p.

ROSADO, J.. Nova Geração de Ferramentas ETL. Disponível na Internet http://www.sybase.pt/gvsview/gvs/sybase-pt/eventos/WorkshopBI2006/ETL.pdf 28 abr. 08

ANEXO A

Neste anexo são mostradas as principais características e estágios da ferramenta *DataStage* 7.5.

A ferramenta é dividida em cinco componentes: *DataStage Administrator, DataStage Designer, DataStage Director, DataStage Manager* e *DataStage Server.*

O *DataStage Server* mantém o repositório de *metadados*, armazena os parâmetros de processos ETL, estabelece conexões com fontes e alvos de dados e realiza efetivamente o processo de extração, transformação e carga dos dados (Servidor). Os outros componentes são apresentados a seguir.

DataStage Administrator

O componente *DataStage Administrator* é utilizado para adicionar, remover ou configurar propriedades de um projeto. Através dele é possível associar privilégios para grupos de usuários do ambiente com dois tipos de funções: *Operator e Developer*.

Usuários do grupo *Operator* podem executar Jobs, que são processos realizados na ferramenta *DataStage* para construir as aplicações envolvendo suas funções, utilizando o componente *DataStage Director*, mas não podem editá-los. Já usuários do grupo *Developer* podem criar e editar os *jobs*.

Ao abrir o *DataStage Administrator,* tem-se três abas principais: *General, Projects* e *Licensing*. Abaixo estão as explicações sobre cada aba.

- Na aba *General* encontra-se a versão do produto e uma opção *Inactivity timeout,* que é utilizada para definir um tempo para que o *DataStage* torne-se inativo se não houver nenhuma ação no tempo determinado.
- Na aba *Projects* são criados os projetos, onde são realizados e organizados os processos de ETL. Os projetos são associados a um diretório que deve ser definido pelo usuário. A figura A1 mostra esta aba.

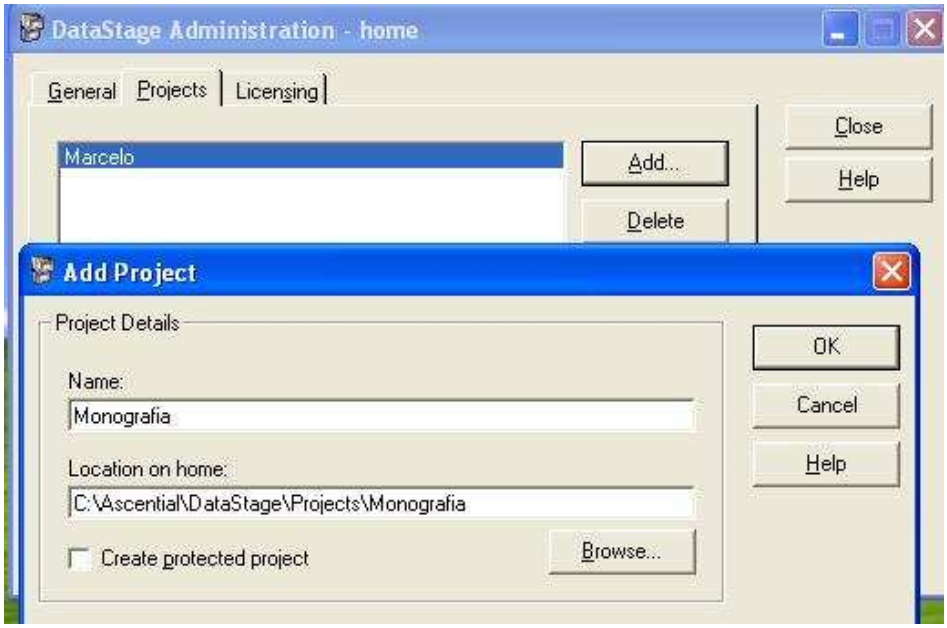


Figura A1 – *DataStage Administrator*

A figura A1 mostra o componente *DataStage Administrator*. Neste componente são encontrados as abas *General*, *Projects* e *Licensing.* Na aba *Projects*, ao clicar em *Add* uma nova janela é aberta para que seja informado o nome do projeto que se deseja criar e sua localização. Por padrão, o *DataStage* já possui um diretório onde os projetos são criados, mas o usuário pode informar a localização desejada.

- Na aba *Licensing* é possível alterar e fazer atualizações referentes às licenças do *DataStage.*

DataStage Manager

O componente *DataStage Manager* atua na manipulação de metadados do *DataStage.* Através dele pode-se importar ou exportar Jobs e definições de formatos dos dados.

A figura A2 mostra a interface do *DataStage Manager.*

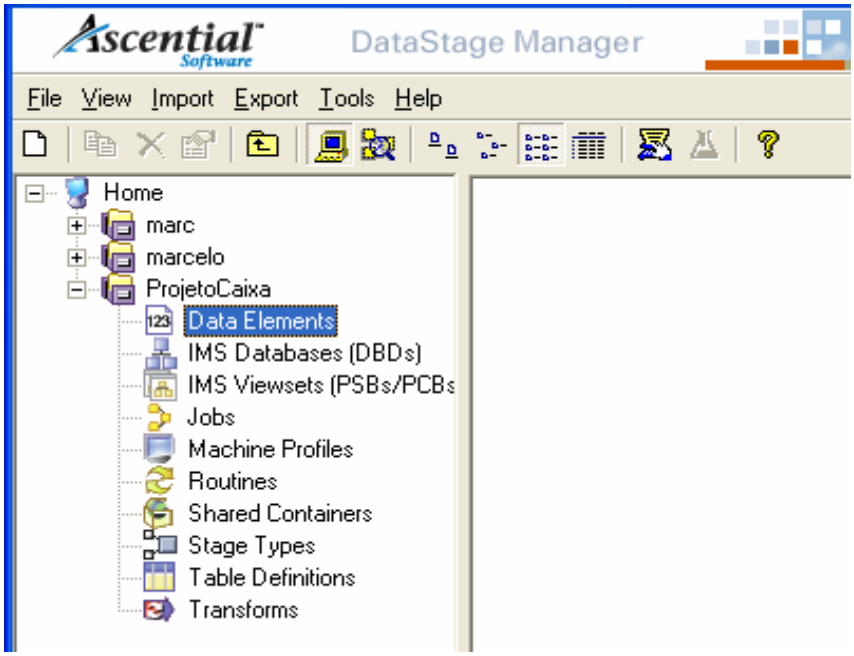


Figura A2 – DataStage Manager

A figura A2 mostra os projetos existentes no *DataStage*. Cada projeto possui opções como: "*Jobs"* onde são listados todos os *Jobs* existentes no projeto; e a opção *Table Definitions*, que armazena os metadados de cada tabela. A opção *Import* é utilizada para importar *Jobs* e componentes criados pelo DataStage de um lugar específico para dentro de um projeto. A opção *Export* é utilizada para exportar componentes criados pelo *DataStage* para um local especificado pelo usuário. Esta última opção é útil para manter *backups* do projeto.

DataStage Director

O componente *DataStage Director* permite validar, executar e monitorar *jobs* compilados pelo *DataStage Designer*. Com ele é possível visualizar o status do *job* em relação à compilação, validação e execução. É possível visualizar *logs* de execução de cada *job*, facilitando a identificação de erros.

A figura A3 apresenta a tela principal do *DataStage Director*.

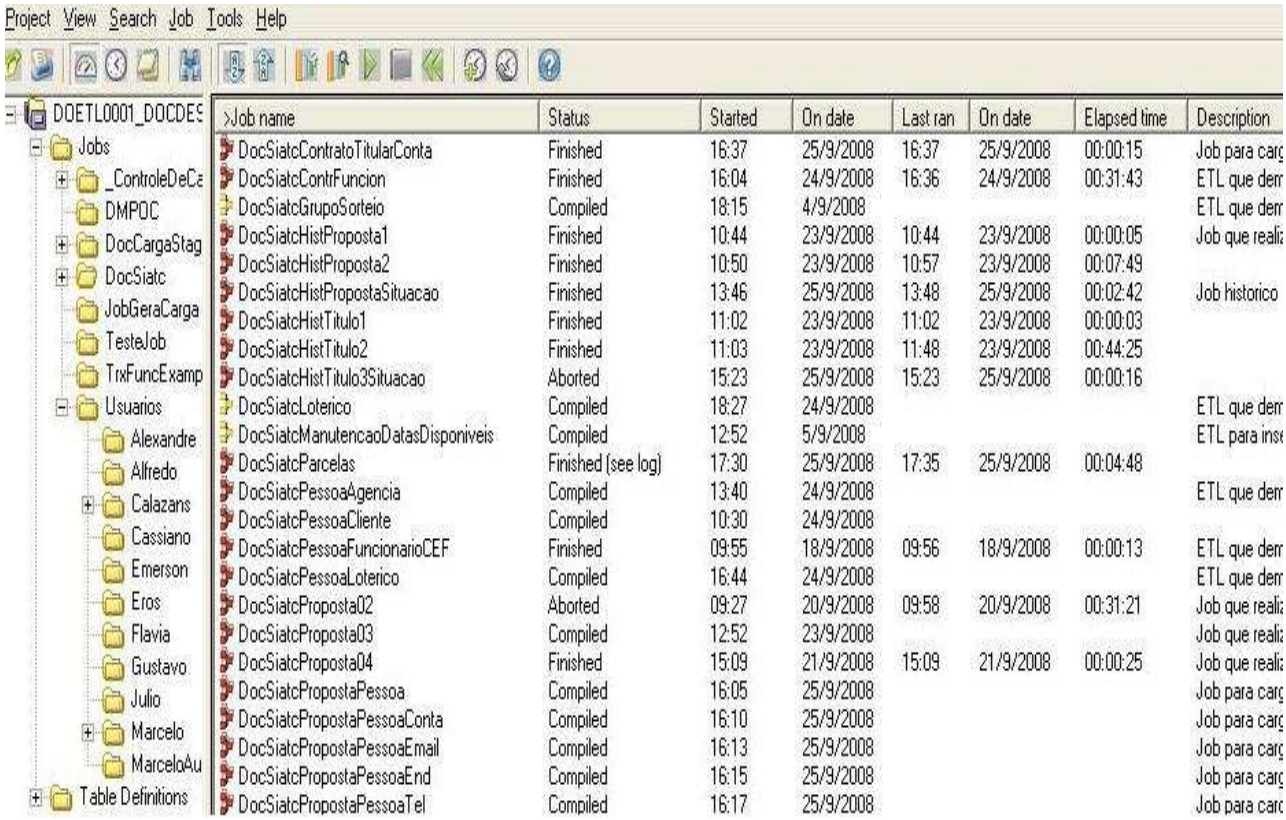
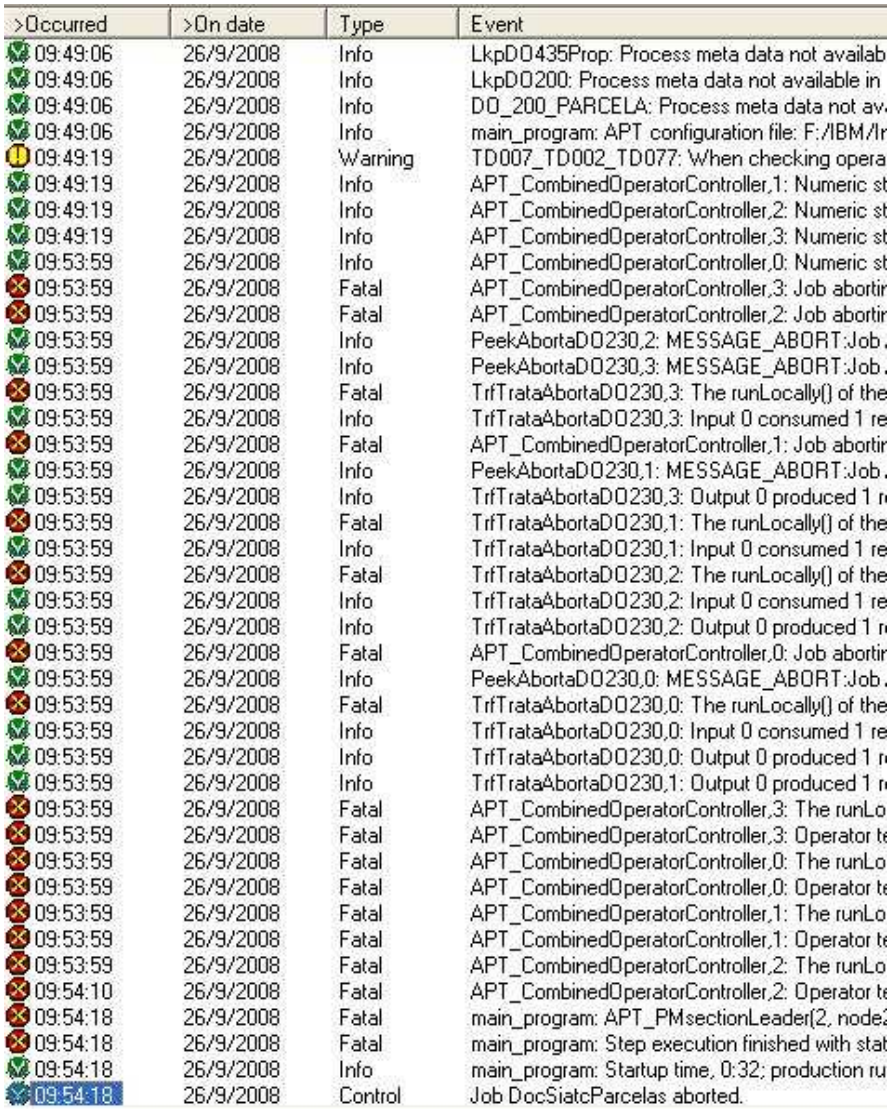


Figura A3 – *DataStage Director*

A figura A3 mostra o componente *DataStage Director*. Do lado esquerdo é apresentado as pastas onde são encontrados os *jobs* presente no projeto. Do lado direito são apresentados todos os *jobs* da pasta selecionada, com algumas informações como data da última vez que foi rodado, quanto tempo durou, e seu estado atual que pode ser: *Compiled, Finished, Finished (see log)* e *Abort.* A primeira opção diz que o *job* foi compilado com sucesso, mas ainda não foi rodado. A segunda diz que foi compilado e executado com sucesso. A terceira informa que o *job* foi rodado, mas gerou *warnings*. E a última diz que o *job* foi abortado porque ao rodar foi gerado erros “fatais”.

Ao clicar em um *job* com o botão direito e selecionar a opção *View Log* é apresentado um *log* com todas as informações sobre o *job*. O *log* pode apresentar três tipos de informação que são identificadas por cores diferentes. Informações de controle são apresentadas com um sinal verde do lado esquerdo. Informações sobre erros que podem comprometer o resultado final são apresentados com um sinal amarelo do lado esquerdo da mensagem informando qual o erro aconteceu e, se possível, sua localização. Estes erros são chamados de *warning*. E Informações sobre erros que comprometem a execução dos *jobs* são chamados de *Fatal Error* e são marcados com um sinal vermelho ao lado da mensagem. Uma

ocorrência de um *Fatal Error* faz com que o *job* aborte. A figura A4 ilustra a tela de *log* de um *job.*



>Occurred	>On date	Type	Event
09:49:06	26/9/2008	Info	LkpDO435Prop: Process meta data not availab
09:49:06	26/9/2008	Info	LkpDO200: Process meta data not available in
09:49:06	26/9/2008	Info	DO_200_PARCELA: Process meta data not av
09:49:06	26/9/2008	Info	main_program: APT configuration file: F:/IBM/I
09:49:19	26/9/2008	Warning	TD007_TD002_TD077: When checking opera
09:49:19	26/9/2008	Info	APT_CombinedOperatorController,1: Numeric st
09:49:19	26/9/2008	Info	APT_CombinedOperatorController,2: Numeric st
09:49:19	26/9/2008	Info	APT_CombinedOperatorController,3: Numeric st
09:53:59	26/9/2008	Info	APT_CombinedOperatorController,0: Numeric st
09:53:59	26/9/2008	Fatal	APT_CombinedOperatorController,3: Job abortir
09:53:59	26/9/2008	Fatal	APT_CombinedOperatorController,2: Job abortir
09:53:59	26/9/2008	Info	PeekAbortaDO230,2: MESSAGE_ABORT:Job
09:53:59	26/9/2008	Info	PeekAbortaDO230,3: MESSAGE_ABORT:Job
09:53:59	26/9/2008	Fatal	TrfTrataAbortaDO230,3: The runLocally() of the
09:53:59	26/9/2008	Info	TrfTrataAbortaDO230,3: Input 0 consumed 1 re
09:53:59	26/9/2008	Fatal	APT_CombinedOperatorController,1: Job abortir
09:53:59	26/9/2008	Info	PeekAbortaDO230,1: MESSAGE_ABORT:Job
09:53:59	26/9/2008	Info	TrfTrataAbortaDO230,3: Output 0 produced 1 r
09:53:59	26/9/2008	Fatal	TrfTrataAbortaDO230,1: The runLocally() of the
09:53:59	26/9/2008	Info	TrfTrataAbortaDO230,1: Input 0 consumed 1 re
09:53:59	26/9/2008	Fatal	TrfTrataAbortaDO230,2: The runLocally() of the
09:53:59	26/9/2008	Info	TrfTrataAbortaDO230,2: Input 0 consumed 1 re
09:53:59	26/9/2008	Info	TrfTrataAbortaDO230,2: Output 0 produced 1 r
09:53:59	26/9/2008	Fatal	APT_CombinedOperatorController,0: Job abortir
09:53:59	26/9/2008	Info	PeekAbortaDO230,0: MESSAGE_ABORT:Job
09:53:59	26/9/2008	Fatal	TrfTrataAbortaDO230,0: The runLocally() of the
09:53:59	26/9/2008	Info	TrfTrataAbortaDO230,0: Input 0 consumed 1 re
09:53:59	26/9/2008	Info	TrfTrataAbortaDO230,0: Output 0 produced 1 r
09:53:59	26/9/2008	Info	TrfTrataAbortaDO230,1: Output 0 produced 1 r
09:53:59	26/9/2008	Fatal	APT_CombinedOperatorController,3: The runLo
09:53:59	26/9/2008	Fatal	APT_CombinedOperatorController,3: Operator te
09:53:59	26/9/2008	Fatal	APT_CombinedOperatorController,0: The runLo
09:53:59	26/9/2008	Fatal	APT_CombinedOperatorController,0: Operator te
09:53:59	26/9/2008	Fatal	APT_CombinedOperatorController,1: The runLo
09:53:59	26/9/2008	Fatal	APT_CombinedOperatorController,1: Operator te
09:53:59	26/9/2008	Fatal	APT_CombinedOperatorController,2: The runLo
09:54:10	26/9/2008	Fatal	APT_CombinedOperatorController,2: Operator te
09:54:18	26/9/2008	Fatal	main_program: APT_PMsectionLeader(2, node2
09:54:18	26/9/2008	Fatal	main_program: Step execution finished with stat
09:54:18	26/9/2008	Info	main_program: Startup time, 0:32; production ru
09:54:18	26/9/2008	Control	Job DocSiatcParcelas aborted.

Figura A4 – *Log* de um *job*

A figura A4 mostra um *log* gerado por um *job.* No *log* é possível obter informações como a hora e a data que ocorreu cada evento do *job* e informações para controlar sua execução como erros fatais e *warnings* e uma descrição detalhada de cada evento.

DataStage Designer

O objetivo do *DataStage Designer* é modelar um fluxo de dados utilizando uma visualização gráfica. Este fluxo é criado editando as propriedades dos estágios e ligações para realizar o processamento necessário. Para logar no *DataStage Designer* é necessário entrar com nome ou número IP do servidor a ser acessado, o nome do usuário, a senha e selecionar o projeto desejado.

A figura A5 apresenta esta tela de conexão.

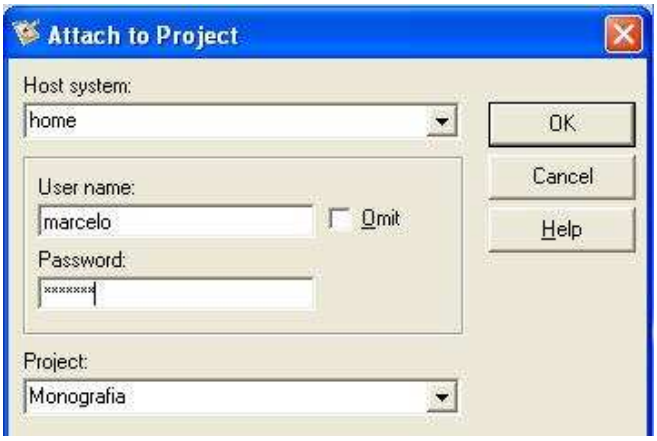


Figura A5 – Conexão com o *DataStage Designer*

Após conectado no *DataStage Designer* é possível iniciar a construção dos *jobs*. A área de trabalho do *DataStage Designer* é destinada ao desenvolvimento dos fluxos dos processos de ETL. Na barra de estágios encontram-se os estágios disponíveis para o desenvolvimento dos *jobs*. A figura A6 mostra a interface do *DataStage Designer*.

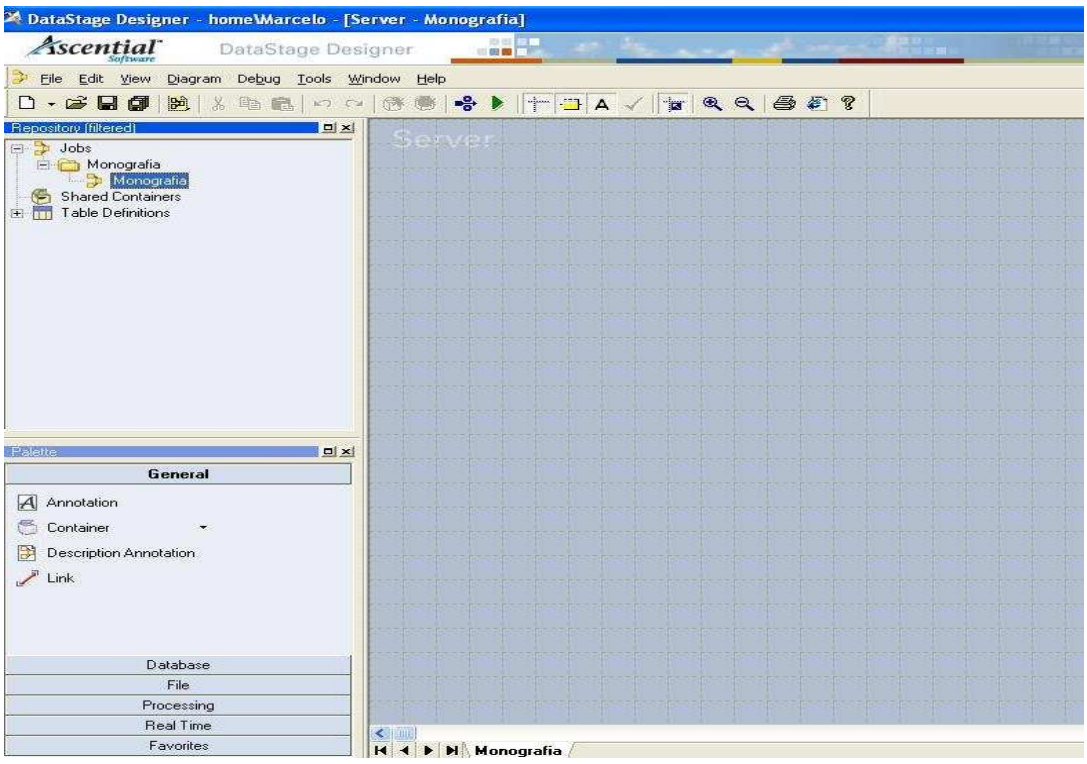


Figura A6 – *DataStage Designer*

A figura A6 mostra a área de desenvolvimento dos *jobs*. O lado direito é o local destinado a construção dos *jobs*. No lado esquerdo, em *Repository*, são encontrados os *jobs* e as *Table Definitions*. Em *Palette* são encontrados os estágios que são utilizados para criar o processo de ETL. A figura A7 apresenta os principais estágios para o desenvolvimento.

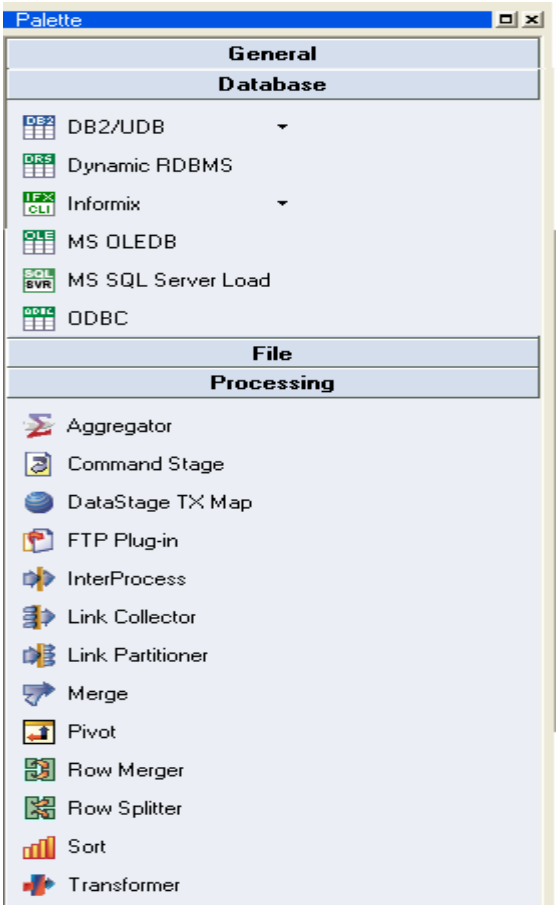


Figura A7 – Estágios do DataStage Designer

A figura A7 apresenta alguns dos estágios que são utilizados para o desenvolvimento do processo de ETL. Estes estágios são separados por grupos: *General*, *Database*, *File* e *Processing*.

Em *General*, os principais estágios são *Annotation* que é utilizado para adicionar um comentário ao *job* e *Link* que é utilizado para criar uma ligação de um estágio para outro. Este *link* é responsável por transportar dados de um estágio para outro.

Em *Database*, encontram-se os estágios nativos dos bancos de dados.

Em *File* encontram-se tipos de estágio para extrair ou carregar arquivos no formato texto. As principais opções são *Sequential File* e *Hashed File* que são apresentadas com mais detalhes mais a frente.

Em *Processing* é onde se localizam os estágios que realizam as transformações dos dados. Há vários estágios, entre eles podemos citar o *Transformer*, *Sort* e *Aggregator*, que são apresentados neste anexo.

Estágio *Sequential File*

O estágio *Sequential File* é utilizado para extrair ou carregar dados em um arquivo no formato texto. Três abas são possíveis de ser encontradas neste estágio.

A primeira é a aba *Stage*, onde é possível definir um nome para o estágio e uma descrição para o estágio. A segunda aba, *Inputs*, só ocorre quando o estágio recebe um *link* de entrada. E a terceira aba, *Outputs*, ocorre quando o estágio possui um *link* de saída. As abas *Inputs* e *Outputs* possuem três subabas: *General*, *Format* e *Collumns*. As funções dessas subabas são semelhantes tanto para o *link* de entrada quanto para o de saída.

Na aba *General* é necessário colocar o caminho com o nome do arquivo texto que será criado e colocar uma descrição desejável. Na aba *Inputs* é possível definir uma ação para o arquivo, que pode ser sobrescrever ou acrescentar os dados e definir se será criado um *backup* do arquivo.

Através da aba *Format* é possível definir se as colunas terão um tamanho fixo, se a primeira linha será o nome das colunas, definir o delimitador que é o responsável por dividir as colunas.

Na aba *Columns*, é possível verificar as colunas presentes no estágio com seu tipo e tamanho, é possível salvar a definição desse arquivo texto para que se possa ser utilizado em outro estágio. Nesta aba, tem um botão chamado *View Data* que ao ser acionado exibe uma tela apresentando os dados contidos no arquivo.

A figura A8 apresenta este estágio.

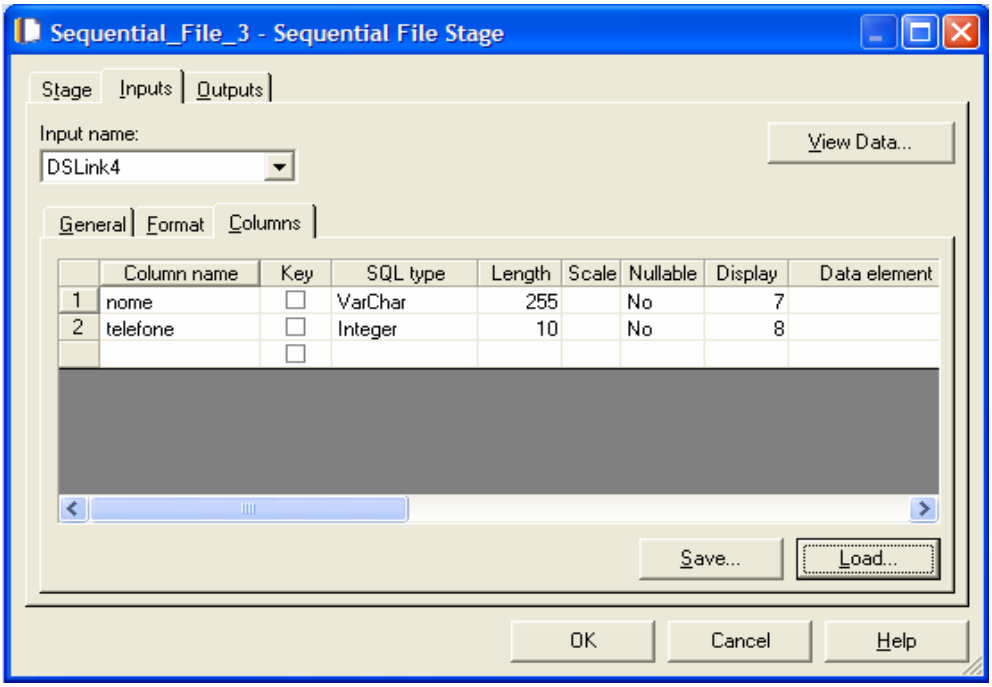


Figura A8 – Estágio *Sequential File*

Estágio *Sort*

O estágio *Sort* é utilizado para ordenar dados. Possui três abas: *Stage*, *Inputs* e *Outputs*.

Em *Stage*, entre opções disponíveis pode-se atribuir o nome para o estágio, sua descrição e definir o tipo de ordenação desejada.

Em *Inputs* e em *Outputs* é possível definir uma descrição ao estágio e verificar as definições das colunas dos dados.

A figura A9 mostra o estágio *Sort.*

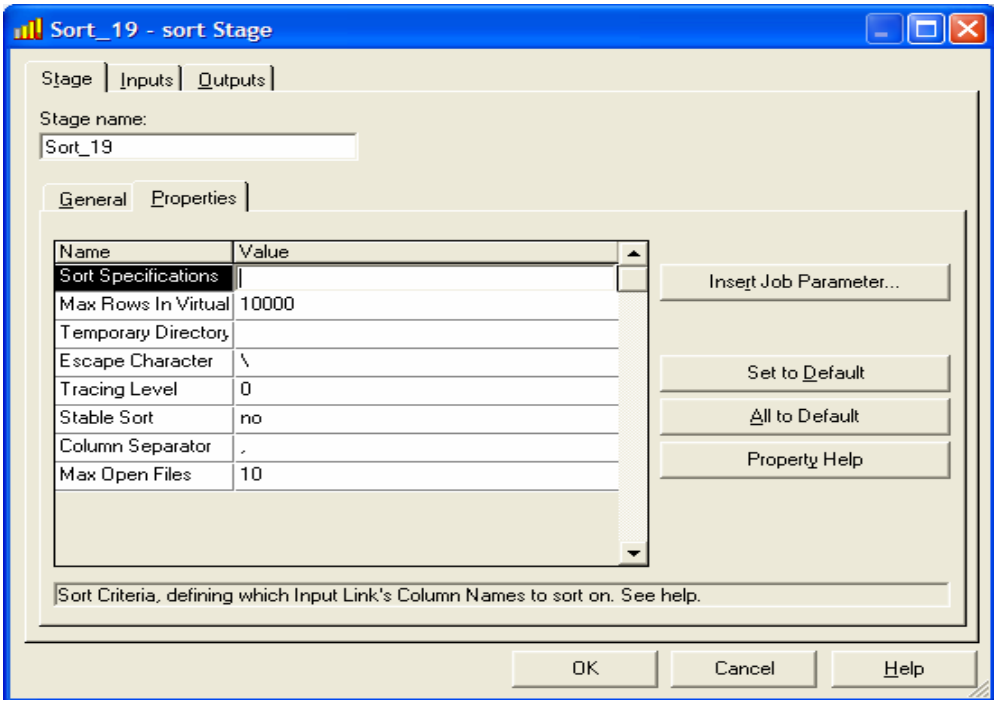


Figura A9 – Estágio Sort

A figura A9 mostra a interface do estágio *Sort*. Na aba *Stage* encontra-se o nome do estágio e na propriedade *Sort Specifications* é onde se deve escolher o tipo de ordenação dos dados. Para utilizar esta propriedade é necessário colocar o nome do campo a ser ordenado seguido de *asc* ou *des* para ascendente ou descendente, respectivamente.

Estágio *Aggregator*

O Estágio *Aggregator* é o responsável por classificar as linhas de dados de um único *link* de entrada em grupos e realizar cálculos em cima desses dados. Semelhante aos estágios apresentados anteriormente, possui três abas principais: *Stage, Inputs* e *Outputs*.

A aba *Stage* apresenta o nome do estágio e uma descrição sobre o mesmo. A aba *Inputs* apresenta uma descrição e a definição das colunas do *link* de entrada.

Na aba *Outputs* é possível agregar os dados e realizar operações em cima destes. É possível retornar o valor máximo ou mínimo de uma coluna, retornar o primeiro ou último registro, sumarizar os dados, calcular valor médio, realizar uma contagem do número de linhas que uma coluna possui ou retornar a divergência padrão para os valores da coluna.

Estas opções podem ser observadas na figura A10.

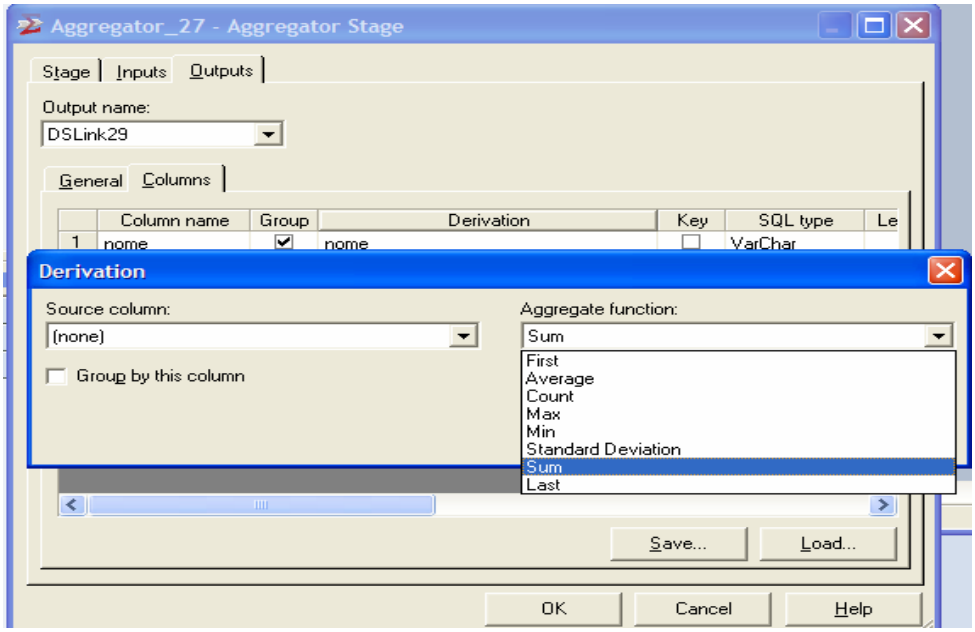


Figura A10 – Estágio Aggregator

A figura A10 mostra a interface do estágio *Aggregator*. Na aba *Outputs* encontram-se todas as colunas que devem seguir para o *link* de saída. Nestas colunas é possível realizar uma *Derivation*, que é o tipo de agregação desejada. Para isto deve-se selecionar uma coluna e aplicar o tipo de agregação correta.

Estágio *Transformer*

O estágio *Transformer* é utilizado para manusear dados extraídos, executar conversões necessárias e fazer *lookups* entre tabelas. O *Transformer* pode possuir vários *links* de entrada e de saída. Através dele pode-se criar ou apagar colunas, alterar a seqüência das colunas, editar os metadados, definir derivações de colunas de saída, unir duas ou mais fonte de dados, criar variáveis para serem utilizadas para controlar algum dado. É possível também criar filtros para selecionar apenas os dados desejados.

A figura A11 apresenta o *Transformer*.

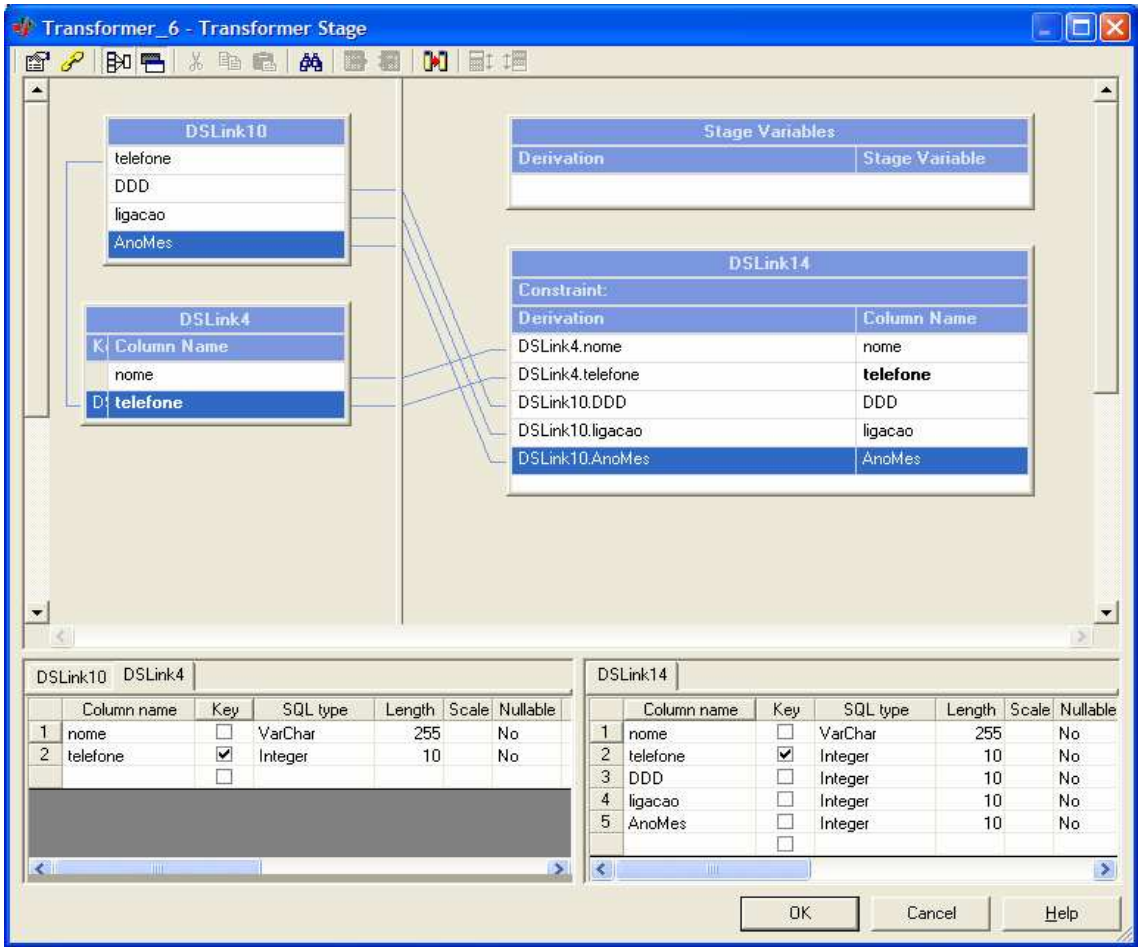


Figura A11 – Estágio Transformer

A figura A11 mostra a interface gráfica do estágio *Transformer*. Ao lado esquerdo é feito um *join* entre dois arquivos, representados pelo nome de *"DSLink10"* e "DSLink4". Como chave para a realização do *join* foi utilizado o campo telefone. O *link* de saída é representado pelo nome "DSLink14" e possui o nome e telefone, obtidos do *link* de entrada "DSLink4", "DDD", "ligação" e "AnoMes" obtidos do *link* de entrada "DSLink10". Na parte inferior é possível ver as colunas e seus metadados.