UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MARCOS VENICIUS UBIALI

CRIAÇÃO DE PROCESSO DE BUSINESS INTELLIGENCE PARA EMPRESA DE SOFTWARE

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2016

MARCOS VENICIUS UBIALI

CRIAÇÃO DE PROCESSO DE BUSINESS INTELLIGENCE PARA EMPRESA DE SOFTWARE

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Profa. Beatriz T. Borsoi

PATO BRANCO
2016

ATA Nº: **284**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO MARCOS VENICIUS UBIALI**.**

Às 13:30 hrs do dia 15 de dezembro de 2016, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Vinicius Pegorini (Convidado) e Viviane Dal Molin de Souza (Convidada), para avaliar o Trabalho de Diplomação do aluno Marcos Venicius Ubiali, matrícula 1295632, sob o título **Criação de Processo de Business Intelligence para Empresa de Software**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 14:05 hrs foi encerrada a sessão.

_____
Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

_____
Prof. Vinicius Pegorini, M.Sc.
Convidado

_____
Profa. Viviane Dal Molin de Souza, M.Sc.
Convidada

_____ _____
Profa. Eliane Maria de Bortoli Fávero, M.Sc — Coordenadora do Trabalho de Diplomação
Prof. Edilson Pontarolo, Dr. — Coordenador do Curso

A Folha de Aprovação assinada encontra-se na Coordenação do Curso

RESUMO

UBIALI, Marcos Venicius. Criação de processo de Business Intelligence para empresa software. 2016. 68f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2016.

Os processos de *Business Intelligence* (BI) possibilitam extrair e comparar dados de diferentes fontes, visualizar informações complexas em tempo real, obter análise de tendências e criar visões com vários níveis de detalhes. Esses dados podem estar armazenados em bases de dados de aplicações distintas e são provenientes de processos de negócio da empresa. Aplicar técnicas de BI sobre esses dados é descobrir informações que, na maioria das vezes, não são obtidas pelas consultas e relatórios oferecidos pelos sistemas de gestão utilizados pela empresa que geram e utilizam esses dados. Em processos de BI dados de fontes distintas podem ser agregados e "minerados" de maneira que informações coletadas desse processo resultam em indicadores que podem tornar-se ferramentas de apoio à tomada de decisão. Este trabalho apresenta a criação de um processo de BI para a extração dos dados de uma aplicação em IBM Smartcloud Control Desk, outra IBM Rational Team Concert e de um sistema desenvolvido pela própria Empresa para uso interno, que é um Call Center. E a partir desses dados gerar indicadores. Esses dados geram um Data Warehouse que foi produzido utilizando as ferramentas e tecnologias Vistra BI, Pentaho Data Integration - Kettle e JavaScript.

Palavras-chave: Integração de bases de dados. Data Warehouse. Business Intelligence. Vistra BI. Pentaho Data Integration.

ABSTRACT

UBIALI, Marcos Venicius. Creation process of Business Intelligence for software company. 2016. 68f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2016.

The processes of Business Intelligence (BI) make possible to extract and compare data from different sources, visualize complex information in real time, get analysis of trends and create visions with various levels of detail. These data are stored in other applications databases and come from the company's business processes. Apply BI techniques on these data is to find that information, in most cases, not available to the query and reporting offered by management systems used by the company and that generate and manipulate data. In BI data from sources other processes can be aggregated and "mined" so that information collected in this process result in indicators that can become support tools for decision making. This work, presenting the creation of a BI process for the extraction of data from IBM SmartCloud Control Desk applications, IBM Rational Team Concert and a system developed by the Company for internal use, which is a Call Center, and from these data to generate indicators. These data generate a Data Warehouse that was produced using the tools and technologies Vistra BI, Pentaho Data Integration - Kettle and JavaScript.

Keywords: Integration of data base. Data Warehouse. Business Intelligence. Vistra BI. Pentaho Data Integration.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

BI	*Business Inltelligence*
DER	Diagrama de Entidades e Relacionamentos
ETL	*Extraction, Transformation, and Loadin*g
ERP	*Enterprise Resource Planning*
FTP	*File Transfer Protocol*
JDBC	*Java Database Connectivity*
OLAP	*Online Analytical Processing*
PDI	*Pentaho Data Integration*
REST	*Representational State Transference*
RGB	*Red Green Blue*
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*
WI	*Work_Items*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais do trabalho, com os seus objetivos e a justificativa. Por fim está a apresentação dos capítulos subsequentes que compõem o texto.

1.1 CONSIDERAÇÕES INICIAIS

A criação de um processo de *Business Intelligence* (BI) pode se tornar necessária por vários motivos, como, por exemplo, para extrair e comparar dados de diferentes fontes, visualizar informações complexas em tempo real, obter análise de tendências e criar visões com vários níveis de detalhe. As informações coletadas em processos de BI resultam em indicadores que podem tornar-se ferramentas de apoio para a tomada de decisão.

A Empresa[1] deseja utilizar as seguintes aplicações para criar um processo de BI:

a) Call Center – sistema desenvolvido pela própria Empresa para uso interno. Esse sistema é utilizado para registrar, atender e gerenciar qualquer tipo solicitação – de cliente ou interna – de todos os setores da Empresa.

b) IBM *Smartcloud Control Desk* – sistema desenvolvido pela IBM com intuito de proporcionar gestão de serviços. A Empresa utiliza esse sistema para controlar as atividades do setor de suporte, como o registro de solicitações dos clientes e os *logs* de atendimento.

c) IBM *Rational Team Concert* – solução de gestão de configuração e mudança desenvolvida pela IBM. A Empresa utiliza essa solução para controle das atividades do setor de fábrica de software. Tais como, as atividades dos desenvolvedores, dos arquitetos, dos analistas de requisitos e dos testadores e o controle de todos os artefatos gerados durante o ciclo de vida do software.

Dados dessas aplicações serão extraídos por meio de um processo de *Extract, Transform and Load* (ETL) e armazenados em um *Data Warehouse* e com isso produzir indicadores. Os indicadores serão utilizados pela área organizacional da empresa, como os coordenadores, gerentes e diretores. Esses indicadores deverão apresentar uma visão da qualidade e da produtividade da empresa.

[1] Como forma de manter a privacidade e preservação de dados, não será identificada a empresa para a qual o processo de BI será criado. Neste texto será tratada como Empresa.

1.2 OBJETIVOS

A seguir são apresentados o objetivo geral e os objetivos específicos deste trabalho.

1.2.1 Objetivo Geral

Criar um processo de BI para a extração dos dados das aplicações IBM *Smartcloud Control Desk*, IBM *Rational Team Concert* e de um sistema desenvolvido pela própria Empresa para uso interno, que é um *Call Center*, e a partir desses dados gerar indicadores.

1.2.2 Objetivos Específicos

- Criar a estrutura de um *Data Warehouse* de modo que seja possível armazenar as informações que serão extraídas, com possibilidade de consulta para outros sistemas diferentes dos envolvidos no processo de BI;
- Desenvolver um processo de ETL que faça a extração dos dados das aplicações IBM *Smartcloud Control Desk*, IBM *Rational Team Concert* e um sistema de *Call Center* desenvolvido pela própria empresa para uso interno;
- Gerar indicadores que representem qualidade e produtividade da empresa, servindo como ferramenta de apoio à decisão.

1.3 JUSTIFICATIVA

Na Empresa para qual o processo de BI será criado há varias ferramentas para controle das atividades em diferentes setores. Entretanto, essa diversidade de ferramentas dificulta a geração de métricas e indicadores a partir desses dados que estão em repositórios distintos. O IBM *Rational Team Concert* é umas delas e por mais que tenha suporte à criação de consultas e gráficos, esses recursos não suprem a demanda dos gestores da Empresa, pois possuem pouca customização e não há forma de integrar ou cruzar dados com as outras ferramentas (IBM *Smartcloud Control* e *Call Center*). O processo de BI proposto como resultado da realização deste trabalho visa resolver esse problema, pois, dessa forma, os dados das

aplicações IBM *Smartcloud Control Desk*, IBM *Rational Team Concert* e um sistema desenvolvido pela própria Empresa para uso interno, um *Call Center*, passam a ser cruzados e disponibilizados apenas em uma base de dados, um *Data Warehouse*.

O *Data Warehouse* desenvolvido a partir de dados dessas três bases, tornará mais simples e ágil o acesso a esse cruzamento de informação, sendo possível gerar os indicadores necessários aos gestores e permitindo o acesso a essas informações com facilidade de consulta e análise.

A programação realizada para o processo de BI é basicamente a escrita de instruções *Structured Query Language* (SQL), tanto na ferramenta de ETL, quanto na ferramenta responsável por prover indicadores. A ferramenta de ETL é o *Pentaho Data Integration* (PDI), nela é definida a estrutura de extração dos dados obtidos das aplicações. Na ferramenta ETL, além de SQL, é necessário codificar instruções em JavaScript para tratamento de *strings*. Para a criação de indicadores é utilizada a ferramenta Vistra BI. Essa é uma solução desenvolvida pela Vistra que é composta pela junção do Vistra DEV, plataforma de desenvolvimento em que pode ser configurada e permite personalizar as análises, pelo Vistra PRO que é uma licença própria para a visualização e manuseio das análises arquitetadas no Vistra DEV. Para complementar a codificação nessa ferramenta, as instruções são desenvolvidas também em SQL.

1.4 ESTRUTURA DO TRABALHO

A sequência deste texto está organizada em capítulos. No Capítulo 2 é apresentado o referencial teórico que fundamenta as tecnologias e os recursos utilizados na realização deste trabalho que é a integração de bases de dados visando obter dados para *Data Warehouse*. O Capítulo 3 apresenta as ferramentas e as tecnologias utilizadas, bem como o método (as atividades) que são passos, em resumo, para agrupar os dados das bases distintas e gerar os indicadoes. Os resultados da realização deste trabalho são apresentados no Capítulo 4. As considerações finais estão no Capítulo 5, seguidas das referências utilizadas para a elaboração do texto.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico do trabalho que está relacionado à *Data Warehouse*.

2.1 DATA WAREHOUSE

Data Warehouse é uma coleção de dados não voláteis, variantes no tempo, integrados, compreensíveis e orientados a assunto que são processados para produzir informação útil e significativa para suportar o gerenciamento de determinada estratégia de decisão (CHAPMAN, 2005). A fim de suportar processos de tomada de decisão, dados no *Data Warehouse* são organizados por assuntos (*subjects*), tais como cliente, item e atividade (CUZZOCREA; PUGLISI, 2011).

Um dos objetivos primários de um *Data Warehouse* é incrementar a inteligência de um processo de decisão e o conhecimento das pessoas envolvidas nesse processo (KANTARDZIC, 2011). Por exemplo, a habilidade de executivos de *marketing* observar um mesmo produto de múltiplas dimensões de desempenho de venda - como por região, tipo de venda, demografia dos consumidores - pode permitir melhores esforços promocionais, incremento de produção ou novas decisões em termos de quantidades em estoque e estratégias de distribuição de produtos. Para Mohajir e Latrache (2012) o objetivo de usar um *Data Warehouse* é ter uma estrutura eficiente para gerenciar informação e analisar dados.

Um *Data Warehouse* possui significado diferente para visões diferentes. Algumas definições são limitadas a dados; outras se referem às pessoas, processos, software, ferramentas e dados. A Oracle (2015) conceitua *Data Warehouse* como uma base de dados relacional que é projetada para pesquisa e análise ao invés de processamento de transações. Esses repositórios geralmente contêm dados históricos derivados de transações de dados, mas podem incluir dados de outras origens.

Assim, um *Data Warehouse* pode ser visto como um repositório de dados da organização, que visa fornecer suporte para a tomada de decisão estratégica. Esses repositórios, geralmente, armazenam enormes quantidades de dados. Uma organização pode ter *Data Warehouses* departamentais frequentemente chamados *Data Marts*. Um *Data Mart* é um *Data Warehouse* que é projetado para encontrar as necessidades de um grupo específico

de usuários. Esses repositórios podem ser grandes ou pequenos, dependendo da área em questão (KANTARDZIC, 2011).

Além de ser uma base de dados relacional, um ambiente de *Data Warehouse* inclui uma solução para extração, transporte, transformação e carga (ETL), uma máquina de processamento analítico *online* (*Online Analytical Processing* (OLAP)), ferramentas cliente de análise e outras aplicações para gerenciamento de processo de obtenção de dados e entrega para usuários de negócio (ORACLE, 2015). ETL representa os processos principais que alimentam um *Data Warehouse* (CUZZOCREA; PUGLISI, 2011). A Figura 1 apresenta uma representação do processo de ETL na alimentacao de *Data Warehouse.*

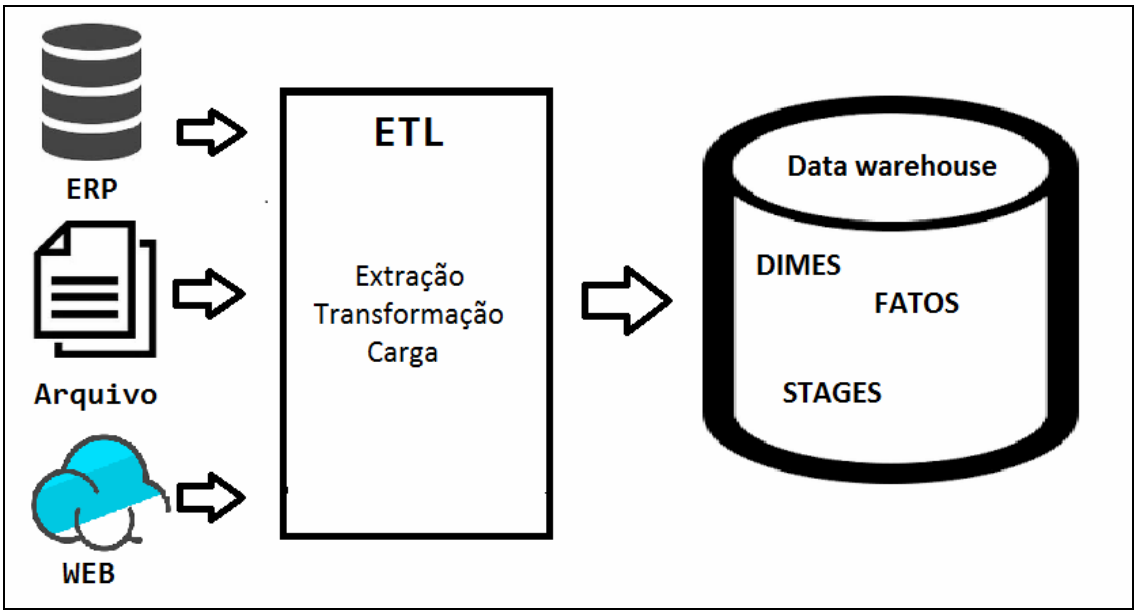

Figura 1 – Representação do processo de ETL na alimentação do Data Warehouse

Para transformar os dados de diferentes fontes e formatos (dados primários, sumarizados e metadados, por exemplo), há quatro tipos principais de transformação e cada um deles possui características próprias (KANTARDZIC, 2011):

a) Transformações simples – essas transformações são os blocos de construção de todas as outras transformações mais complexas. Essa categoria inclui manipulação de dados que estão em um único campo por vez, sem levar em conta seus valores em campos relacionados.

b) Limpeza – essa transformação assegura formatação consistente e uso de um campo ou grupos de campos relacionados. Essa classe de transformação também inclui verificações para validar valores em um determinado campo, geralmente verificando a faixa de valor ou escolhendo de uma lista enumerada.

c) Integração – dados de fontes diversas são mapeados campo a campo em uma nova estrutura de dados em um *Data Warehouse*.

d) Agregação e sumarização – são métodos para condensar instâncias de dados encontrados em ambientes operacionais em uma quantidade menor de instâncias em um ambiente de *Data Warehouse*.

Savitri e Laksmiwati (2011) definem o curso principal de desenvolvimento de um *Data Warehouse* como processo de extração, limpeza, transformação e carga. Para esses autores, o processo de extração está relacionado a obter os dados de entrada para o *Data Warehouse*. A limpeza envolve uma série de processos para eliminar dados indesejados, provavelmente causados por inconsistências, duplicidade, redundância, dados inválidos e perdidos. Esse é o processo de transformação que formata dados do *Data Warehouse* para que possam ser utilizados. O processo carga está relacionado à transferência dos dados obtidos das fontes de origem para serem armazenados no sistema de *Data Warehouse*.

A modelagem de Data Warehouse é feita utilizando modelos dimensionais, de forma que, seja extremamente simples, fazendo com que a consulta seja facilitada e rápida, sem necessidade de consultar várias tabelas para acessar a informação desejada. Os modelos dimensionais mais aceitos são:

a) Star schema – modelo recomendado para Data Warehouse de todos os tamanhos. As tabelas DIME se relacionam com as tabelas FATOS. Ocupa menos espaço, mas esse modelo tende a ter tabelas DIMES maiores.

b) Snow flake - modelo recomendado para Data Warehouse de pequeno porte, assim como no modelo *star schema*, as tabelas DIME se relacionam com as tabelas FATOS. Contudo, nesse modelo tabelas DIME podem relacionar-se entre si, especializando ou normalizando ainda mais as tabelas DIME. Ocupa mais espaço, diminui a complexidade das tabelas DIME, mas aumenta a complexidade das consultas por haver mais tabelas envolvidas.

Na modelagem de Data Warehouse tem dois tipos principais de tabelas, que são elas:

a) DIMES - são as tabelas que armazenam informações descritivas, que fornecem perspectiva às tabelas fatos.

b) FATOS – são as tabelas que fornecem as métricas, valores e quantidades, no geral, atributos numéricos que representam informações quantitativas, quando cruzadas com as tabelas DIMES, geram informações relevantes que podem ser usadas em análises.

3 MATERIAIS E MÉTODO

Neste capítulo são apresentadas as ferramentas e as tecnologias utilizadas e as atividades realizadas para o desenvolvimento do trabalho.

3.1 MATERIAIS

Para a realização da integração das três bases de dados referenciadas nos objetivos deste trabalho serão utilizadas as ferramentas e as tecnologias apresentadas no Quadro 1.

Ferramenta / Tecnologia	**Versão**	**Referência**	**Finalidade**
IBM Smart Cloud Control Desk	7.5.1.1	http://www-03.ibm.com/software/products/pt/control-desk	Fornecer os dados de gestão de serviços que serão integrados.
IBM Rational Team Concert	5.0.2	https://jazz.net/	Fornecer os dados de gestão de configuração e mudança que serão integrados.
Call Center		Ferramenta interna da empresa	Fornecer os dados internos de atendimentos que serão integrados.
Vistra BI		http://www.vistra.com.br/solucoes/vistra-bi/	Ferramenta para construção de indicadores.
Pentaho Data Integration - Kettle	4.3.0	http://community.pentaho.com/projects/data-integration/	Ferramenta de ETL.
JavaScript		https://www.javascript.com/	Para composição de instruções na ferramenta ETL.
SQL		http://www.w3schools.com/sql/	Para composição das instruções para consulta de dados na ferramenta ETL e Vistra BI.
IBM DB2 LUW Express-C	10.5	http://www.ibm.com/analytics/us/en/technology/db2/db2-trials.html	Sistema Gerenciador de Banco de Dados (SGBD) utilizado como Data Warehouse.

Quadro 1 - Tecnologias e ferramenta utilizadas no desenvolvimento do trabalho

A seguir são apresentadas as principais tecnologias utilizadas, constantes no Quadro 1.

a) IBM *Smart Cloud Control Desk* é um sistema desenvolvido pela IBM com intuito de proporcionar gestão de serviços. A Empresa utiliza esse sistema para controlar as

atividades do setor de suporte, como registro de solicitações dos clientes e *logs* de atendimento.

b) IBM *Rational Team Concert* é uma solução de gestão de configuração e mudança desenvolvida pela IBM. A Empresa utiliza essa solução para controle das atividades do setor de fábrica de software que são realizadas por desenvolvedores, arquitetos, analistas de requisitos e testadores e, também, para o gerenciamento de todos os artefatos gerados durante o ciclo de vida do software.

c) *Call Center* é um aplicativo desenvolvido pela própria Empresa para uso interno. Esse aplicativo é utilizado para registrar, atender e gerenciar qualquer tipo solicitação (cliente ou interna) de todos os setores da empresa.

d) Vistra BI é uma plataforma desenvolvida pela Vistra que é composta pela junção do Vistra DEV, plataforma de desenvolvimento que com o uso do Vistra PRO permite configurações e personalizações de análises. O Vistra PRO é uma licença própria para a visualização e manuseio das análises arquitetadas no Vistra DEV.

e) Pentaho *Data Integration* ou Kettle é uma ferramenta de ETL desenvolvida em Java e é *open source* sob a licença Apache License 2.0. Suporta conexão com mais de 20 bancos de dados, realiza leitura e escrita de arquivos, além de chamadas a *web services* e requisições *Representational State Transference* (REST).

f) JavaScript é uma linguagem de programação baseada em *script*.

g) SQL é a linguagem de programação utilizada para interagir com bancos de dados relacionais.

h) IBM DB2 LUW Express-C é um SGBD. A sigla LUW significa que atende ambientes Linux, Unix e Windows. A letra C significa que é a versão da comunidade, sem custo, mas apresenta restrições em relação à versão não comunitária.

3.2 MÉTODO

A seguir são apresentados os principais passos realizados para a criação do processo de BI para a Empresa.

3.2.1 MAPEAMENTO DA ORIGEM DOS DADOS

O acesso às informações dos sistemas IBM *Smartcloud Control Desk* e *Call Center* foi feita diretamente nos bancos de dados desses sistemas, por meio de conexão nativa *Java Database Connectivity* (JDBC), utilizando de instruções SQL do tipo SELECT.

O IBM *Rational Team Concert* também possui um banco de dados IBM DB2 com possibilidade de acesso por meio de conexão nativa JDBC. Entretanto, esse banco de dados não é modelado de forma relacional e pela ausência de documentação que explique a localização de gravação e a forma de organização das informações, não foi possível utilizar esse recurso. Outra tentativa foi realizada por meio da API REST (JAZZ, 2016) que é disponibilizada pela própria *Rational Team Concert*, mas pelo fato de não haver documentação clara de como autenticar a requisição para esse tipo de serviço via Kettle, não foi possível utilizar esse recurso.

A solução mais simples encontrada foi utilizar o recurso de consulta do IBM *Rational Team Concert*. A Empresa já fazia uso de algumas consultas para analisar determinadas informações. Os dados dessas consultas eram exportados em planilhas Excel para o formato *csv* e eram utilizadas para montar indicadores, na maioria dos casos, gráficos. Visto que os dados relevantes já estavam em planilhas Excel e a Kettle suporta leitura desse tipo de arquivos, foi dado sequência na criação do processo de BI fazendo a integração dessas planilhas com as informações dos outros sistemas.

3.2.1.1 RELEVÂNCIA DE DADOS EM IBM RATIONAL TEAM CONCERT

As informações relevantes do IBM Rational Team Concert são as atividades (work_items ou WI) cadastradas pelo setor da fábrica da Empresa. Somente os WI como defeitos, *bugs* e auditorias foram considerados. O Quadro 2 mostra as informações que estavam nas planilhas das consultas do IBM *Rational Team Concert*. Essas informações vêm da planilha *Enterprise Resource Planning* (ERP) e de frente de caixa e foram extraídas do IBM *Smartcloud Control Desk.*

Informação	Explicação
AREADEPROJETO	Projeto onde esta alocada o wi
ARQUIVADOEM	Equipe onde esta alocada a wi
CHILDREN	Lista de wis filhas dessa wi
CLIENTE	Cliente origem da wi
CRIADOPOR	Nome da pessoa que cadastrou o wi
DTCRIACAO	Data de criação
DTRESOLUCAO	Data de resolução
ENCONTRADOEMPRODUCAO	Em caso de ser um defeito, especifica se foi encontrado na versão em produção
ESTIMATIVA	Estimativa para desenvolvimento
WI_ID	Numero de identificação do wi
IDCONTROLDESK	Numero de identificação da solicitação de serviço que gerou o wi
LOCALIZADOEM	Em que versão esta
MODULO	Descrição do modulo origem
PARENT	Numero do wi pai que gerou este wi
PLANEJADOPARA	Versão para qual o wi esta planejado
PRIORIDADE	Prioridade do wi
PROPRIEDADEDE	Responsável pelo wi
RELEASELIST	Lista de versões para quais esse wi será liberado
RESOLUCAO	Quando resolvido(cancelado ou concluído) especifica o motivo (pronto, cancelado ou duplicado)
RESOLVIDOPOR	Nome da pessoa que resolveu o wi
RESUMO	Resumo do wi
STATUS	Especifica se o desenvolvimento do wi esta em andamento, cancelar ou resolver
TEMPOGASTO	Tempo gasto para desenvolver o wi
TIPO	Tipo do wi (Defeito e Bug)

Quadro 2 - Informações da planilha ERP e frente de caixa

No Quadro 3 estão informações obtidas da planilha GCO que também foram extraídas do IBM Smartcloud Control Desk.

Informação	Explicação
CRIADOPOR	Número da solicitação de serviço
DTCRIACAO	Data de alteração da solicitação de serviço
ENCONTRADOEMPRODUCAO	Especifica se a auditoria foi encontrada na versão em produção
EQUIPEORIGEM	Equipe origem da auditoria
WI_ID	Numero da auditoria (wi)
MODULO	Modulo origem da auditoria
ITEMMODULO	Item de modulo origem da auditoria
PROPRIEDADEDE	Responsável pela auditoria
PROGRAMADOR	Programador origem da auditoria
RELEASEORIGEM	Versão origem da auditoria
DTRESOLUCAO	Data de resolução
SITUACAO	Quando encerrada especifica a situação.
STATUS	Especifica se o desenvolvimento do wi esta em andamento, cancelar ou resolver
RESUMO	Resumo da auditoria
TESTADOR	Testador origem da auditoria
TIPO	Tipo do wi (auditoria)

Quadro 3 - Informações da planilha GCO

3.2.1.2 RELEVÂNCIA DE DADOS NO IBM SMARTCLOUD CONTROL DESK

As informações relevantes do IBM *Smartcloud Control Desk* são as solicitações de serviço que são cadastradas pelo setor de suporte da empresa. Contudo, apenas as solicitações consideradas como 'Erro de Sistema' (defeitos) e que tenham sido analisadas pelo comitê que encaminha as solicitações ao setor de fábrica, foram extraídas. O Quadro 4 mostra as informações que foram extraídas.

Informação	Descrição
TICKETID	Número da solicitação de serviço
OWNDATE	Data de alteração da solicitação de serviço

Quadro 4 - Dados relevantes extraídos do IBM Smartcloud Control Desk

O critério aplicado para extração dos dados do Quadro 3 é apresentado no Quadro 5.

Informação	Operador	Valor
OWNER	IN(...)	'GESTAOMUDANCAS' 'GMUDANCA'
TIPODEMANDA	=	'Erro de Sistema'

Quadro 5 - Critério aplicado para extração dos dados do IBM Smartcloud Control Desk

3.2.1.3 RELEVÂNCIA DE DADOS DO CALL CENTER

As informações relevantes do sistema Call Center são as solicitações de atendimento cadastradas pelo setor de suporte da empresa e pelas atividades cadastradas pelo setor da fábrica como "Erro de Programa" (defeitos). O Quadro 6 mostra as informações que foram extraídas do sistema de Call Center.

Informação	Explicação
IDCLIENTE	Código do cliente
NOMECLIENTE	Nome do cliente
IDCHAMADO	Código da solicitação de atendimento
IDMODULO	Código do módulo
DESCRMODULO	Descrição do módulo
IDITEMMODULO	Código do item de módulo
DESCRITEMMODULO	Descrição do item de módulo
IDCATEGORIA	Código da categoria
DESCRCATEGORIA	Descrição da categoria
DTCADASTRO	Data de cadastro da solicitação de atendimento
DTRESOLUCAO	Data de resolução da solicitação de atendimento
IDSITUACAO	Código da situação
DESCRSITUACAO	Descrição da situação
TIPO	Solicitação de atendimento para defeito
ORIGEM	Origem interna ou de cliente

Quadro 6 - Dados relevantes extraídos do Call Center

No Quadro 7 é apresentado o critério aplicado para extração de dados das solicitações de atendimento cadastradas pelo setor de suporte da empresa e das atividades pelo setor da fábrica como erro de programa.

Informação	Operador	Valor
IDCATEGORIA	=	2

Quadro 7 - Critério aplicado para extração dos dados do Call Center

3.2.2 LEVANTAMENTO DOS REQUISITOS

As planilhas com as informações do IBM Rational Team Concert devem ser importadas na integra, com ressalva aos dados que devem ter transformados antes da gravação, conforme evidenciado na Seção 3.2.4 DESENVOLVIMENTO DO PROCESSO DE ETL, conteúdo dos importDimeWorkItemGco.ktr e importDimeWorkItems.ktr.

As informações do IBM Rational Team Concert e Call Center devem estar organizadas de forma analítica no banco de dados para que outros sistemas se beneficiem da unificação e da organização dessas informações.

A cada execução do processo de ETL, os dados devem ser excluídos e gerados novamente.

A periodicidade da execução do processo de ETL não foi um ponto comentado durante o levantamento de requisitos. Como trata-se de um projeto pequeno, pelo menos inicialmente, não haveria problemas de desempenho se a execução do todo o processo de ETL fosse executada diariamente. Contudo, conforme o crescimento da massa de dados, recomenda-se a avaliação da periodicidade da execução.

3.2.3 DEFINIÇÃO DO BANCO DE DADOS DATA WAREHOUSE

Após o levantamento dos requisitos e do mapeamento da origem dos dados, é possível fazer a modelagem das tabelas do *Data Warehouse*. A seguir são apresentados os padrões de nomenclatura utilizados e o Diagrama de Entidades e Relacionamentos (DER) das tabelas do *Data Warehouse*.

a) Tabelas com o prefixo DIME armazenam informações cadastrais, como por exemplo: DIME_VERSOES mantém todos os códigos de versão do sistema.

b) O prefixo STAGE do início do nome indica que a tabela armazena os dados da forma mais bruta possível após a extração. Dessa forma, eliminando ao máximo, regras e tratamentos que possam contribuir para a demora da extração.

c) Tabelas FATO são as tabelas que mantêm os dados prontos, tratados, com regras de negócio aplicadas, ou seja, a informação que será mostrada no indicador. As regras, os tratamentos e os *joins*, são feitos no ETL, de forma que, no VISTRA a instrução SQL do indicador, seja o mais simples possível, focando na velocidade de acesso à informação.

O Diagrama de Entidades e Relacionamentos das tabelas do *Data Warehouse* é apresentado na Figura 2.

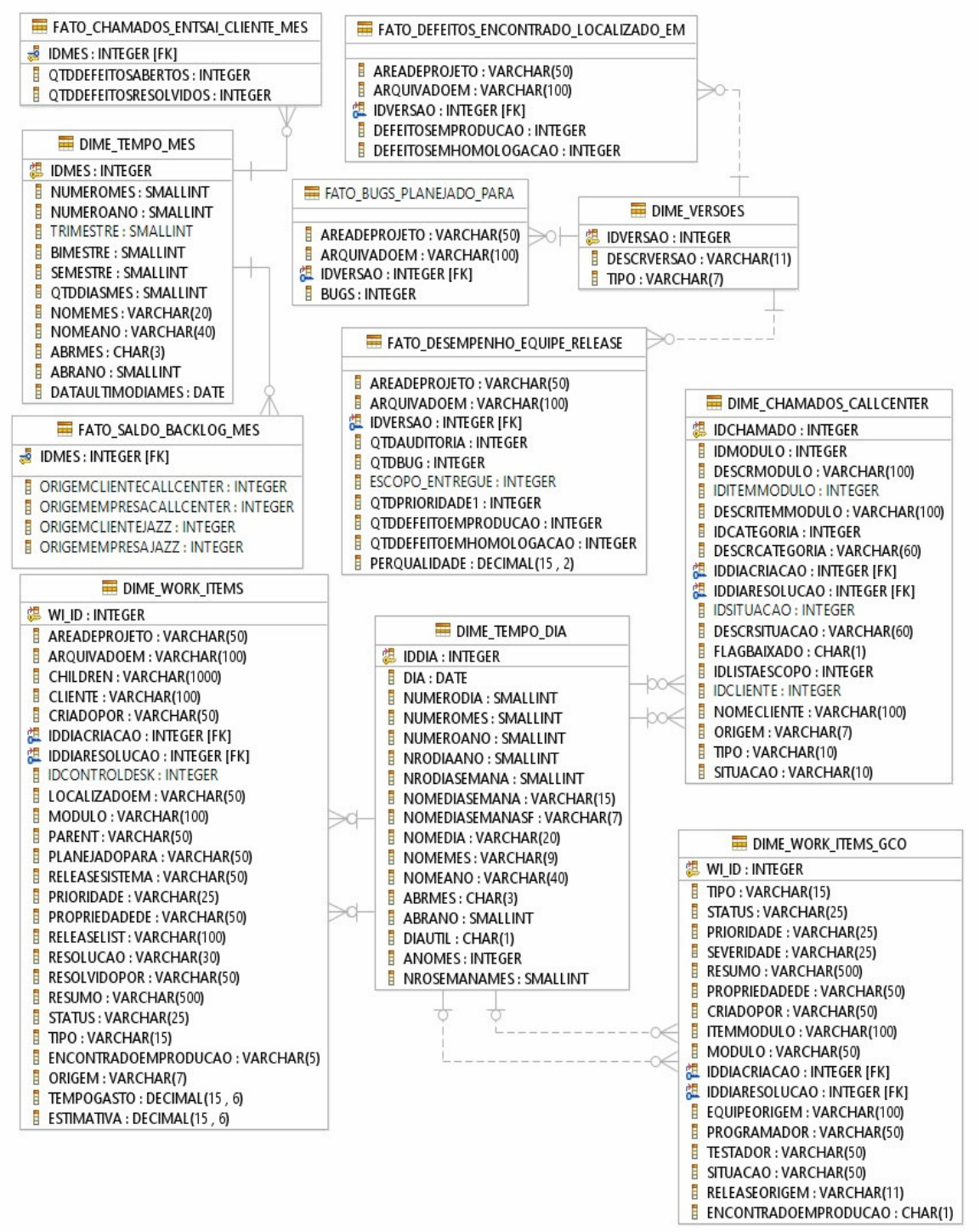

Figura 2 - Diagrama de entidades e relacionamentos do *Data Warehouse*

3.2.4 DESENVOLVIMENTO DO PROCESSO DE ETL

A criação do processo de ETL na Kettle é feita utilizando-se de *jobs* (.kjb) e transformations (.ktr), esses ligados por linhas (hops) que podem ser de falha (em cor vermelha), sucesso (em cor verde) ou que ignora o resultado e prossegue com a execução (cor preta). O conteúdo de *job* pode ser executado em paralelo, mas o conteúdo de um transformation não. Um *job* pode conter transformations, *jobs* e *steps*, mas um transformation contém apenas *steps*. Os *steps* dentro de um transformation servem para mover e transformar dados de uma fonte para outra. Já os *steps* dentro de *jobs* executam funções mais complexas e específicas, como, aplicar condições, enviar e-mails, comunicar-se com *File Transfer Protocol* (FTP) e gerenciar arquivos dentro do sistema operacional.

Na pasta que está a Kettle foi criada uma pasta fontes e dentro dessa as seguintes pastas: dimes, dimesnv2, fatos, fatosnv2.

Os seguintes componentes foram desenvolvidos:

a) Componentes da pasta fontes: manager.ktr;

b) Componentes da pasta dimes: dimes.kjb, dimeChamadosCall Center.ktr, dimeWorkItems.ktr e dimeWorkItemsGco.ktr;

c) Componentes da pasta dimesvn2: dimesnv2.kjb, dimeVersoes.ktr e dimeWorkItemsVersaoSaida.ktr;

d) Componentes da pasta fatos: fatos.kjb, fatoBugsPlanejadoPara.ktr, fatoChamadosEntSaiClienteMes.ktr e fatoDefeitosEncontradoLocalizadoEm.ktr;

e) Componentes da pasta fatosnv2: fatosnv2.kjb, fatoDesempenhoEquipeRelease.ktr e fatoSaldoBacklogMes.ktr;

As seguintes conexões JDBC foram definidas com as bases de dados envolvidas no processo de ETL:

a) JDBCBIFAB para o *Data warehouse*;

b) JDBCSD apontada para o IBM *Smartcloud Control Desk*;

c) JDBCCALLCENTER configurado para o *Call Center*.

Após definidas no manager.kjb, essas conexões devem ser compartilhadas para que os jobs, transformations e steps tenham a visibilidade de conexão.

Dentro do manager.kjb, cada *step* e *job* são ligados por um hop verde, mas cada job deve ter um hop vermelho ligado a um step Write To Log, de modo que, caso ocorra algum erro, uma mensagem deve ser escrita no log de execução da Kettle.

Em seguida é necessário definir as seguintes configurações e adicionar os seguintes *steps* e *jobs*:

a) *Step START*. É obrigatório haver esse *step* em um *job*, a execução dentro de um *job* inicia nesse *step*.

b) *Step Set Variables*. *Step* utilizado para definir variáveis e seu escopo. No Set Variables definir uma variável com a localização dos arquivos necessários exportados do IBM Rational Team Concert, a variável deve ter visibilidade em toda máquina virtual Java.

c) Step Set Check Db connections. Step utilizado para testar se é possível conectar o banco de dados definido em uma conexão, caso haja falha a execução deve ser interrompida, utilizar step Abort job para isso. Adicionar as três conexões definidas no job manager. Adicionar os jobs, stages, dimes, fatos e fatosnv2. Cada um desses deve estar ligado a um step Write To Log, por meio de um hop vermelho, de modo que, caso ocorra algum erro, uma mensagem deve ser escrita no log de execução da Kettle.

d) Step Success. Step no qual termina o processo de ETL dentro do job manager.

A Figura 3, no Capítulo Resultado, mostra como deve ficar após realizadas essas configurações.

Dentro do dimes.kjb, os transformations devem ser executados em paralelo. Essa execução vai seguir para o step Sucess se a execução do transformation ocorreu com sucesso, caso contrário executar um step Write To Log, de modo que, caso ocorra algum erro, uma mensagem deve ser escrita no log de execução da Kettle.

e) Step START. É obrigatório haver esse step em um job. A execução dentro de um job inicia neste step.

É necessário adicionar os componentes dimeChamadosCall Center.ktr, importDimeWorkItemGco.ktr e importDimeWorkItems.ktr

f) Step Success. Step no qual termina o processo de ETL dentro do job manager.

A Figura 4, no Capítulo 4, mostra o resultado após a execução desses steps.

Dentro do dimeChamadosCall Center.ktr a execução deve iniciar com a limpeza dos dados da tabela DIME_CHAMADOS_CALL CENTER e apenas quando este terminar, será a feita a carga com as novas informações.

g) Step Execute SQL script, executa uma ou várias instruções SQL. Em connection definir JDBCBIFAB e em script adicionar a instrução SQL da Listagem 4.

h) Step Block this step until steps finish, bloqueia a execução dos steps até o término da execução de um terceiro step. Configurar para aguardar a execução do step Execute SQL script.

i) Step Table input, executa uma instrução SQL do tipo SELECT no banco de dados e envia o retorno da instrução para os steps posteriores. Em connection é necessário definir JDBCCALLCENTER e em script adicionar a instrução SQL que está na Listaagem 4.

j) Step Table output, recebe informações e executa uma instrução SQL do tipo INSERT no banco dados. Em connection, target schema e target table definir respectivamente JDBCBIFAB, DBA e DIME_CHAMADOS_CALL CENTER. A Figura 5, no Capítulo 4, apresenta a imagem da ferramenta após a execução desse step.

Dentro do importDimeWorkItemGco.ktr a execução deve iniciar com a limpeza dos dados da tabela DIME_WORK_ITEMS_GCO e apenas quando este terminar, será a feita a carga com as novas informações.

k) Step Execute SQL script, em connection definir JDBCBIFAB e em script adicionar a instrução SQL que está na Listagem 4.

l) Step Block é necessário configurar para aguardar a execução do step Execute SQL script.

m) Step Microsoft Excel Input, realiza a leitura de planilhas Excel. As informações lidas são passadas para os steps posteriores. Na Aba files opção selected files adicionar uma linha, em File/Directory inserir ${VAR_PATH_PLANILHAS} e em Wildcard (RegExp) inserir .*gco.xlsx. Aba Sheets em list sheets to read adicionar uma linha, em Sheet name inserir gco. Aba Fields com a ajuda do botão Get Fields from Header now adicionar os campos a serem lidos.

n) Step Modified Java Script Value, permite por meio de JavaScript, transformar os dados que estão transitando entre os steps. Em script adicionar o JavaScript que está na Listagem 5. Na opção Fieds adicionar duas linhas, em Fieldname outIDDIACRIACAO, outIDDIARESOLUCAO e em Type, String para todas as linhas. Esse step é necessário para transformar os campos Creation_Date e Resolution_Date que são do tipo date (2016-08-30) para apenas um numero inteiro que representa a data (20160830).

o) Step Table output, em connection, target schema e target table definir respectivamente JDBCBIFAB, DBA e DIME_WORK_ITEMS_GCO. Marcar a opção Specify database fields e na aba Database Fields fazer a associação dos campos da tabela com os campos retornados pelo Step Microsoft Excel Input, para os campos IDDIACRIACAO e IDDIARESOLUCAO utilizar respectivamente as variáveis, outIDDIACRIACAO e outIDDIARESOLUCAO. A representação após a execução desse step é apresentada na Figura 6, que está no Capítulo 4.

Em importDimeWorkItems.ktr a execução deve iniciar com a limpeza dos dados da tabela DIME_WORK_ITEMS e apenas quando esse processo terminar, será a feita a carga com as novas informações.

p) Step Execute SQL script, em connection definir JDBCBIFAB e em script adicionar a instrução SQL da Listagem 6.

q) Step Block this step until steps finish, configurar para aguardar a execução do step Execute SQL script.

r) Step Microsoft Excel Input, realiza a leitura de planilhas do Excel, as informações lidas são passadas para os steps posteriores. Serão necessários dois steps, um para fazer o input das atividades relacionadas ao ERP e outro para as atividades relacionadas ao sistema de frente de caixa.

No Microsoft Excel Input para leitura das atividades relacionadas ao ERP, na aba files opção selected files adicionar uma linha, em File/Directory inserir ${VAR_PATH_PLANILHAS} e em Wildcard (RegExp) inserir .*erp.xlsx. Aba Sheets em list os sheets to read adicionar uma linha, em Sheet name inserir erp. Aba Fields com a ajuda do botão Get Fields from Header now adicionar os campos a serem lidos.

Microsoft Excel Input para leitura das atividades relacionadas ao sistema de frente de caixa: na aba files opção selected files adicionar uma linha, em File/Directory inserir ${VAR_PATH_PLANILHAS} e em Wildcard (RegExp) inserir .*frentecaixa.xlsx. Aba Sheets em list os sheets to read adicionar uma linha, em Sheet name inserir frentecaixa. Aba Fields com a ajuda do botão Get Fields from Header now adicionar os campos a serem lidos.

s) Step Modified Java Script Value, permite por meio de JavaScript transformar os dados que estão transitando entre os steps. Em script adicionar o JavaScript que está na Listagem 7. Na opção Fieds adicionar nove linhas, em Fieldname outRELEASESISTEMA, outRELEASELIST, outPLANEJADOPARA, outPARENT, outIDDIACRIACAO, outIDDIARESOLUCAO, outORIGEM, outESTIMATIVA , outTEMPOGASTO e em Type, String para todas as linhas. Esse step é necessário para fazer as seguintes transformações:

- Campo RELEASESISTEMA, deve ser calculado pelo campo PLANEJADOPARA, retirando qualquer texto que esteja acompanhando a versão do sistema. Exemplo: ‘Build 14.0.5.40 SP1’ deve ficar apenas ‘14.0.5.40’;

- Campos PARENT E RELEASELIST, devem ter o primeiro caractere retirado;

- Campo PLANEJADOPARA deve receber ‘Unassigned’ quando o seu valor for nulo;

- Campos DATACRIACAO e DATARESOLUCAO que são do tipo date devem ser transformados para um número inteiro que representa a data;

- Campo ORIGEM, deve ser calculado por meio do campo CLIENTE, quando o valor for nulo ou diferente de ‘EMPRESA’, o valor do campo deve ser ‘CLIENTE’, caso contrário, ‘EMPRESA’;

- Campos TEMPOGASTO e ESTIMATIVA devem ser transformados em apenas um valor decimal que represente o tempo em segundos.

t) Step Table output, em connection, target schema e target table definir respectivamente JDBCBIFAB, DBA e DIME_WORK_ITEMS. Marcar a opção Specify database fields e na aba Database Fields fazer a associação dos campos da tabela com os campos retornados pelo Step Microsoft Excel Input, para os campos PARENT, PLANEJADOPARA, RELEASELIST, RELEASESISTEMA, IDDIACRIACAO, IDDIARESOLUCAO, ORIGEM, ESTIMATIVA e TEMPOGASTO utilizar respectivamente as variáveis outPARENT, outPLANEJADOPARA, outRELEASELIST, outRELEASESISTEMA, outIDDIACRIACA, outIDDIARESOLUCAO, outARQUIVADOEM, outESTIMATIVA e outTEMPOGASTO.

A Figura 7, no Capítulo 4, apresenta a imagem da ferramenta após a execução desse step.

No dimesnv2.kjb, o fluxo execução dos transformations devem seguir a mesma lógica do dimes.kjb.

u) Step START. É obrigatório haver esse step em um job, a execução dentro de um job inicia nesse step. É necessário adicionar o componente dimeVersoes.ktr.

v) Step Success. Step no qual termina o processo de ETL dentro do job manager. A imagem da ferramenta após a execução desse step é a que consta na Figura 8 que está no Capítulo 4.

Em dimeVersoes.ktr a execução deve iniciar com a limpeza dos dados da tabela DIME_VERSOES e apenas quando este terminar, será a feita a carga com as novas informações.

w) Step Table input, executa uma instrução SQL do tipo SELECT no banco de dados e envia o retorno da instrução para os steps posteriores. Em connection definir JDBCBIFAB e em script adicionar a instrução SQL que está na Listagem 8.

x) Step Insert/Update recebe informações e executa uma pesquisa na tabela utilizando uma ou mais colunas com critério, caso o registro não exista, uma instrução SQL do tipo INSERT é realizada, caso o registro exista, um instrução SQL do tipo UPDATE é executada. Em connection, target schema e target table definir respectivamente JDBCBIFAB, DBA e DIME_VERSOES. Em Keys to look up, IDVERSAO = IDVERSAO e em update fields

adicionar as três colunas da tabela, marcar para não executar update na coluna IDVERSAO. A Figura 9, que está no Capítulo de Resultado, apresenta a imagem após a execução desse step.

Em fatos.kjb, o fluxo execução dos transformations deve seguir a mesma lógica do stages.kjb.

y) Step START. É obrigatório haver esse step em um job, a execução dentro de um job inicia neste step. É necessário, ainda, adicionar os componentes fatoBugsPlanejadoPara.ktr, fatoChamadosEntSaiClienteMes.ktr e fatoDefeitosEncontradoLocalizadoEm.ktr.

z) Step Success. Step no qual termina o processo de ETL dentro do job manager. Na Figura 10, Capítulo 4, está a imagem após a realização desse step.

Dentro do fatoBugsPlanejadoPara.ktr a execução deve iniciar com a limpeza dos dados da tabela FATO_BUGS_PLANEJADO_PARA e apenas quando este terminar, será a feita a carga com as novas informações.

aa) Step Execute SQL script, executa uma ou várias instruções SQL. Em connection definir JDBCBIFAB e em script adicionar a instrução SQL da Listagem 9.

ab) Step Block this step until steps finish, bloqueia a execução dos steps depois dele até o término da execução de um terceiro step. Configurar para aguardar a execução do step Execute SQL script.

ac) Step Table input, executa uma instrução SQL do tipo SELECT no banco de dados e envia o retorno da instrução para os steps posteriores. Em connection definir JDBCBIFAB e em script adicionar a instrução SQL da Listagem 10.

ad) Step Table output, recebe informações e executa uma instrução SQL do tipo INSERT no banco dados. Em connection, target schema e target table definir respectivamente JDBCBIFAB, DBA e FATO_BUGS_PLANEJADO_PARA. A Figura 11, que está no Capítulo 4, apresenta a imagem da ferramenta após a execução desse step.

Em fatoChamadosEntSaiClienteMes.ktr a execução deve iniciar com a limpeza dos dados da tabela FATO_CHAMADOS_ENTSAI_CLIENTE_MES e apenas quando este terminar, será a feita a carga com as novas informações.

ae) Step Execute SQL script, executa uma ou várias instruções SQL. Em connection definir JDBCBIFAB e em script adicionar a instrução SQL (Listagem 11).

af) Step Block this step until steps finish, bloqueia a execução dos steps depois dele até o término da execução de um terceiro step. Configurar para aguardar a execução do step Execute SQL script.

ag) Step Table input, executa uma instrução SQL do tipo SELECT no banco de dados e envia o retorno da instrução para os steps posteriores. Serão necessários dois, um que fará a busca no Data Warehouse e outro que fará a busca no IBM Smartcloud Control Desk.

Table input para busca no Data Warehouse, em connection definir JDBCBIFAB e em script adicionar a instrução SQL da Listagem 12.

Table input para busca no IBM Smartcloud Control Desk, em connection definir JDBCSD e em script adicionar a instrução SQL (Listagem 13).

ah) Step Sort rows, aplica uma ordenação nos dados recebidos de um step, de forma ascendente ou descendente. Serão necessários dois: um que fará a ordenação dos dados retornados do Table input que consultou o Data Warehouse e o outro que fará a ordenação dos dados do Table input que consultou o IBM Smartcloud Control Desk. Para ambos os Sort rows a ordenação deve ser feita pela coluna IDMES de forma ascendente.

ai) Step Merge Join, recebe dados de dois steps diferentes e faz a ligação entre eles utilizando uma coluna como critério de ligação, similar ao JOIN de uma instrução SQL do tipo SELECT. Em First Step selecionar um Sort rows, Second Step selecionar o outro Sort rows, Join Type, right outer. Key field IDMES para os dois steps.

aj) Step If field value is null, valida se o valor recebido é nulo, caso seja, substitui por outro valor. Marcar a opção Select Fields e em Fields adicionar os campos QTDDEFEITOSABERTOS e QTDDEFEITOSRESOLVIDOS com a opção Replace by value com valor 0.

ak) Step Table output, recebe informações e executa uma instrução SQL do tipo INSERT no banco dados. Em connection, target schema e target table definir respectivamente JDBCBIFAB, DBA e FATO_CHAMADOS_ENTSAI_CLIENTE_MES.

A imagem da execução desse step está na Figura 12 (Capítulo 4).

Dentro do fatoDefeitosEncontradoLocalizadoEm.ktr a execução deve iniciar com a limpeza dos dados da tabela FATO_DEFEITOS_ENCONTRADO_LOCALIZADO_EM e apenas quando este terminar, será a feita a carga com as novas informações.

al) Step Execute SQL script, executa uma ou várias instruções SQL. Em connection definir JDBCBIFAB e em script adicionar a instrução SQL da Listagem 14.

am) Step Block this step until steps finish, bloqueia a execução dos steps depois dele até o término da execução de um terceiro step. Configurar para aguardar a execução do step Execute SQL script.

an) Step Table input, executa uma instrução SQL do tipo SELECT no banco de dados e envia o retorno da instrução para os steps posteriores. Em connection definir JDBCBIFAB e em script adicionar a instrução SQL da Listagem 15.

ao) Step Table output, recebe informações e executa uma instrução SQL do tipo INSERT no banco dados. Em connection, target schema e target table definir respectivamente JDBCBIFAB, DBA e FATO_DEFEITOS_ENCONTRADO_LOCALIZADO_EM.

Em fatosnv2.kjb, o fluxo execução dos transformations devem seguir a mesma lógica do stages.kjb

ap) Step START. É obrigatório haver esse step em um job, a execução dentro de um job inicia neste step. É necessário adicionar os componentes fatoDesempenhoEquipeRelease.ktr e fatoSaldoBacklogMes.ktr.

aq) Step Success. Step no qual termina o processo de ETL dentro do job manager.

No Capítulo 4, a Figura 13 apresenta a ferramenta após a execução desse step.

Em fatoDesempenhoEquipeRelease.ktr a execução deve iniciar com a limpeza dos dados da tabela FATO_DESEMPENHO_EQUIPE_RELEASE e apenas quando este terminar, será a feita a carga com as novas informações.

ar) Step Execute SQL script, executa uma ou várias instruções SQL. Em connection definir JDBCBIFAB e em script adicionar a instrução SQL da Listagem 16.

as) Step Block this step until steps finish, bloqueia a execução dos steps depois dele até o termino da execução de um terceiro step. Configurar para aguardar a execução do step Execute SQL script.

at) Step Table input, executa uma instrução SQL do tipo SELECT no banco de dados e envia o retorno da instrução para os steps posteriores. Em connection definir JDBCBIFAB e em script adicionar a instrução SQL (Listagem 17).

au) Step Table output, recebe informações e executa uma instrução SQL do tipo INSERT no banco dados. Em connection, target schema e target table definir respectivamente JDBCBIFAB, DBA e FATO_DESEMPENHO_EQUIPE_RELEASE. A imagem da ferramenta após a execução desse step está na Figura 14 do Capítulo 4.

Dentro do fatoSaldoBacklogMes.ktr a execução deve iniciar com a limpeza dos dados da tabela FATO_SALDO_BACKLOG_MES e apenas quando este terminar, será a feita a carga com as novas informações.

av) Step Execute SQL script, executa uma ou várias instruções SQL. Em connection definir JDBCBIFAB e em script adicionar a instrução SQL da Listagem 18.

aw) Step Block this step until steps finish, bloqueia a execução dos steps depois dele até o término da execução de um terceiro step. Configurar para aguardar a execução do step Execute SQL script.

ax) Step Table input, executa uma instrução SQL do tipo SELECT no banco de dados e envia o retorno da instrução para os steps posteriores. Em connection definir JDBCBIFAB e em script adicionar a instrução SQL que está na Listagem 19.

ay) Step Table output, recebe informações e executa uma instrução SQL do tipo INSERT no banco dados. Em connection, target schema e target table definir respectivamente JDBCBIFAB, DBA e FATO_SALDO_BACKLOG_MES. A imagem da Figura 15, Capítulo 4, apresenta a ferramenta após a execução desse step.

3.2.5 DESENVOLVIMENTO DOS INDICADORES

Para o desenvolvimento dos indicadores foi utilizada a plataforma Vistra BI que é composta pela junção do Vistra DEV, plataforma de desenvolvimento em que se pode configurar e personalizar as análises, pelo Vistra PRO, licença própria para a visualização e manuseio das análises arquitetadas no Vistra DEV. O acesso à plataforma Vistra é feito por meio de um usuário e senha. O usuário precisa ser do perfil DEV e tipo Administrador.

Na plataforma, a criação dos indicadores é feita com o uso de várias ferramentas, mas para a criação dos indicadores propostos, foram utilizadas apenas as seguintes:

a) Consulta SQL. Ferramenta que executa uma instrução SQL no banco de dados definido em sua conexão. É possível definir parâmetros dentro de uma Consulta SQL no momento de execução.

b) Gráfico. É a ferramenta com o maior número de configurações, por isso foram utilizados alguns padrões para o desenvolvimento. A plataforma suporta a criação de vários gráficos dentro da ferramenta Gráfico, no caso de haver mais de um, eles ficam separados por abas. Mas neste trabalho, cada ferramenta Gráfico, contém apenas um gráfico com suas séries e cada série pode executar uma Consulta SQL diferente. A ferramenta suporta a criação de gráficos, sendo eles de barra, ponto, linha, setores (pizza), área e radar. Cada tipo de gráfico habilita ou desabilita um tipo de configuração. Neste trabalho foram utilizados gráficos de barra e de linha.

c) Medidor. Os Medidores são ferramentas que podem apresentar as informações na forma de velocímetros, termômetros ou relógios digitais. Neste trabalho foi utilizado apenas o estilo relógio digital para os medidores.

d) Painel. Os painéis são os indicadores, nos painéis são adicionados os gráficos e os medidores desenvolvidos.

O Quadro 8 apresenta o padrão utilizado para desenvolvimento de ferramentas do tipo Gráfico.

Construção Tipo de gráfico: barra ou linha; Opções de série: habilitar hint para cada série do gráfico, a visualização do hint deve ser valor. **Apresentação** Gráfico, aparência, cor de fundo, cor 244; 244; 244 (padrão *Red Green Blue,* RGB). Estilo de preenchimento, modo de preenchimento, sólido. Borda, desmarcar a opção visível; Painéis, cor de fundo, transparente. Estilo de preenchimento, modo de preenchimento, sólido. Borda, desmarcar a opção visível. Eixos: Eixo X primário, aparência, cor Black. Rótulos, Geral, Configurações de Texto, cor Black. Eixo Y mesmo padrão do X, se o gráfico for de linha, em Elementos, Linhas de Grade, marcar a opção visível. Séries, aparência, cor, definir a cor utilizando o padrão RGB. Estilo de preenchimento, modo de preenchimento, sólido. Borda, desmarcar a opção visível. Se o gráfico for de linha, em marcador, marcar a opção visível. Estilo, tipo círculo, desmarcar a opção exibir borda. Estilo de preenchimento, modo de preenchimento, sólido. A configuração deve ser feita para cada série do gráfico. Rótulo de Pronto, geral, marcar a opção visível. Linhas, se gráfico de barra marcar a opção visível se gráfico de linha desmarcar. Aparência, cor de fundo, transparente. Estilo de preenchimento, modo de preenchimento, sólido. Borda, desmarcar a opção visível. Legendas, se o gráfico é de apenas uma série, em geral, desmarcar a opção visível. Caso tenha mais de uma série, em geral, marcar a opção visível, direção, esquerda para direita. Alinhamento, vertical, fundo fora, horizontal, centro. Aparência, cor de fundo, transparente. Estilo de preenchimento, modo de preenchimento, sólido. Borda, desmarcar a opção visível. Titulo, marcar a opção visível. Formatação, cor, black, fonte tahoma, 14pt, regular, marcar a opção anti-serrilhado.

Quadro 8 - Padrão utilizado para desenvolvimento de ferramentas do tipo gráfico

Configurações iniciais para trabalhar com o Vistra BI

No Vistra BI logar com o usuário criado conforme especificações, acessar Opções, Configurador do Sistema, Conexões. Criar uma nova conexão, definindo um identificador e

SGBD, que são BIFAB e DB2 respectivamente. Dentro dessa conexão, será necessário informar os dados de conexão com o Data Warehouse, que são eles:

Identificador. Exemplo: BIFAB

Formato da data, que deve ser DMY

IP do Servidor do banco de dados. Exemplo: 172.16.106.211

Nome do banco de dados. Exemplo: BIFAB

Porta de acesso ao banco de dados. Exemplo: 53000

Schema no qual as tabelas se encontram. Exemplo: DBA

Usuário. Exemplo: dba

Senha. Exemplo: jucabala

Ainda em Configurador do Sistema, acessar Conexões por Usuário, selecionar a opção por usuário e em conexão selecionar BIFAB. Na tabela do lado esquerdo, selecionar o usuário utilizado para logar na plataforma e na tabela do lado direito, marcar a opção Selecionado para este usuário. Após isso, será necessário deslogar e logar novamente.

Acessar a ferramenta Vistra BI após fazer as configurações iniciais, aba Desenvolvedor, Consultas SQL, Grupos, criar um novo grupo para armazenar as Consultas SQL. Exemplo: INDICADORES FÁBRICA. Para criar uma nova Consulta SQL, acessar a opção Novo e informar:

Informar descrição da consulta. Exemplo: Análise de Defeitos - Gráfico Entrada Clientes x Saída Fábrica;

Grupo Exemplo: INDICADORES FÁBRICA;

Conexão com o banco de dados. Exemplo: BIFAB;

Modo, escrita.

Confirmar, adicionar a instrução SQL do tipo SELECT e salvar. Realizar esse processo para as seguintes Consultas SQL:

a) Análise de Defeitos - Gráfico Evolução Mensal Backlog, utilizar a instrução SQL do tipo SELECT que está na Listagem 20;

b) Análise de Defeitos - Gráfico Entrada Clientes x Saída Fábrica, utilizar a instrução SQL do tipo SELECT (Listagem 21);

c) Análise de Defeitos - Medidores Entrada Clientes x Saída Fábrica, utilizar a instrução SQL do tipo SELECT, na Listagem 22;

d) Evolução de Qualidade - Gráfico % de Qualidade ERP, utilizar a instrução SQL do tipo SELECT da Listagem 23;

e) Evolução de Qualidade - Gráfico % de Qualidade Frente de Caixa, utilizar a instrução SQL do tipo SELECT, na Listagem 24;

Com as Consultas SQL criadas, acessar a aba Desenvolvedor, Gráficos, Grupos, criar um novo grupo para armazenar os Gráficos. Exemplo: INDICADORES FÁBRICA. Para criar uma nova Consulta SQL, acessar a opção Novo e informar:

Informar descrição da consulta. Exemplo: Análise de Defeitos - Entrada Clientes x Saída Fábrica;

Grupo Exemplo: INDICADORES FÁBRICA;

Selecionar a opção confirmar.

Dentro das configurações do gráfico informar o mesmo nome para título de identificação da aba e título do gráfico. Exemplo: Evolução Defeitos Abertos x Resolvidos. Realizar os passos a seguir para série do gráfico: em série, adicionar uma nova, selecionar a consulta que vai prover os dados, informar o nome da série, o campo de valor e o campo de agrupamento, se necessário informar a função (soma, contagem, maior, menor) e definir a ordenação e confirmar.

Criar os gráficos a seguir utilizando a ferramenta Gráfico.

Gráfico Análise de Defeitos - Entrada Clientes x Saída Fábrica

Para os campos título de identificação da aba e título do gráfico informar: Evolução Defeitos Abertos x Resolvidos;

Adicionar séries ao gráfico:

a) Consulta: Análise de Defeitos - Gráfico Entrada Clientes x Saída Fábrica

Nome série: Abertos;

Valor: QTDDEFEITOSABERTOS;

Agrupamento: MES;

Função: Em branco;

Ordenação: Sem ordenação.

b) Consulta: Análise de Defeitos - Gráfico Entrada Clientes x Saída Fábrica

Nome série: Resolvidos

Valor: QTDDEFEITOSRESOLVIDOS

Agrupamento: MES

Função: Em branco;

Ordenação: Sem ordenação.

c) Consulta: Análise de Defeitos - Gráfico Entrada Clientes x Saída Fábrica

Nome série: Média Abertos

Valor: MEDIADEFEITOSABERTOS

Agrupamento: MES

Função: Em branco;

Ordenação: Sem ordenação.

d) Consulta: Análise de Defeitos - Gráfico Entrada Clientes x Saída Fábrica

Nome série: Média Resolvidos

Valor: MEDIADEFEITOSRESOLVIDOS

Agrupamento: MES

Função: Em branco

Ordenação: Sem ordenação.

Gráfico Análise de Defeitos - Evolução Mensal Backlog

Para os campos título de identificação da aba e título do gráfico informar: BackLog de Defeitos;

Adicionar séries ao gráfico:

a) Consulta: Análise de Defeitos - Gráfico Evolução Mensal Backlog

Nome série: Saldo Total;

Valor: BACKLOG;

Agrupamento: MES;

Função: Em branco;

Ordenação: Sem ordenação.

b) Consulta: Análise de Defeitos - Gráfico Evolução Mensal Backlog

Nome série: Saldo Call Center;

Valor: BACKLOGCALL CENTER

Agrupamento: MES

Função: Em branco;

Ordenação: Sem ordenação.

c) Consulta: Análise de Defeitos - Gráfico Evolução Mensal Backlog

Nome série: Saldo Jazz

Valor: BACKLOGJAZZ

Agrupamento: MES

Função: Em branco;

Ordenação: Sem ordenação.

Gráfico Evolução de Qualidade - Gráfico % de Qualidade ERP

Para os campos identificação da aba e título do gráfico informar: % de Qualidade ERP.

Adicionar séries ao gráfico:

a) Consulta: Análise de Defeitos - Gráfico % de Qualidade ERP

Nome série: Meta;

Valor: PERMETAQUALIDADE:

Agrupamento: DESCRVERSAO

Função: Em branco;

Ordenação: Sem ordenação.

b) Consulta: Análise de Defeitos - Gráfico % de Qualidade ERP

Nome série: % de Qualidade;

Valor: PERQUALIDADE;

Agrupamento: DESCRVERSAO;

Função: Em branco;

Ordenação: Sem ordenação.

Gráfico Evolução de Qualidade - Gráfico % de Qualidade Frente de Caixa

Para os campos identificação da aba e título do gráfico informar: BackLog de Defeitos

Adicionar séries ao gráfico:

a) Consulta: Evolução de Qualidade - Gráfico % de Qualidade Frente de Caixa

Nome série: % de Qualidade

Valor: PERQUALIDADE;

Agrupamento: DESCRVERSAO;

Função: Em branco;

Ordenação: Sem ordenação.

b) Consulta: Evolução de Qualidade - Gráfico % de Qualidade Frente de Caixa

Nome série: Meta;

Valor: PERMETAQUALIDADE;

Agrupamento: DESCRVERSAO;

Função: Em branco;

Ordenação: Sem ordenação.

Dentro do gráfico acessar a opção aparência e fazer as configurações conforme descrito no Padrão utilizado para desenvolvimento de ferramentas do tipo Gráfico.

Com as Consultas SQL criadas, acessar a aba Desenvolvedor, Medidores, Grupos, criar um novo grupo para armazenar os medidores. Exemplo: INDICADORES FÁBRICA. Acessar a opção novo e informar:

a) Nome do Medidor. Exemplo: Análise de Defeitos - Entrada Clientes x Saída Fábrica;

Grupo Exemplo: INDICADORES FÁBRICA;

Resultado de. Exemplo: Consulta SQL;

Consulta. Exemplo: Análise de Defeitos - Medidores Entrada Clientes x Saída Fábrica

Campo. Exemplo: PERPONTOEQUILIBRIO

Selecionar a opção confirmar.

Dentro da configuração do Medidor, acessar galeria de estilos, e definir um estilo que seja do tipo digital. Conforme especificações, criar os medidores a seguir:

b) Nome do Medidor: Entrada Clientes x Saída Fábrica - Medidor % Ponto de Equilibrio;

Grupo: INDICADORES FÁBRICA

Resultado de: Consulta SQL;

Consulta: Análise de Defeitos - Medidores Entrada Clientes x Saída Fábrica;

Campo: PERPONTOEQUILIBRIO.

c) Nome do Medidor: Entrada Clientes x Saída Fábrica - Medidor Média Abertos;

Grupo: INDICADORES FÁBRICA

Resultado de: Consulta SQL;

Consulta: Análise de Defeitos - Medidores Entrada Clientes x Saída Fábrica;

Campo MEDIADEFEITOSABERTOS.

d) Nome do Medidor: Entrada Clientes x Saída Fábrica - Medidor Média Resolvidos;

Grupo: INDICADORES FÁBRICA

Resultado de: Consulta SQL;

Consulta: Análise de Defeitos - Medidores Entrada Clientes x Saída Fábrica;

Campo MEDIADEFEITOSRESOLVIDOS.

e) Nome do Medidor: Entrada Clientes x Saída Fábrica - Medidor Qtd Ponto de Equilibrio;

Grupo: INDICADORES FÁBRICA;

Resultado de: Consulta SQL;

Consulta: Análise de Defeitos - Medidores Entrada Clientes x Saída Fábrica;

Campo QTDPONTOEQUILIBRIO.

f) Nome do Medidor: Entrada Clientes x Saída Fábrica - Medidor Saldo;

Grupo: INDICADORES FÁBRICA;

Resultado de: Consulta SQL;

Consulta: Análise de Defeitos - Medidores Entrada Clientes x Saída Fábrica;

Campo SALDO.

Desenvolvido os gráficos, acessar aba Desenvolvedor, Painéis, Grupos, criar um novo grupo para armazenar Painéis. Exemplo: INDICADORES FÁBRICA. Para criar um novo Painel, acessar a opção Novo, em branco e informar:

Nome do Painel. Exemplo: Evolução de Qualidade;

Grupo: INDICADORES FÁBRICA;

Em dimensões em pixels, informar 1269 para largura e 570 para altura.

Selecionar a opção confirmar.

Dentro da ferramenta Painel, para cada abar a ser criada, acessar Opções, abas, configurar e informar:

Nome. Exemplo: % de Qualidade ERP;

Cor de fundo: No botão, informar os valores conforme padrão RGB (220,220,220);

Tema: Office2010Silver.

Painel Evolução de Qualidade

Na aba % de Qualidade ERP, adicionar uma ferramenta do tipo Gráfico, selecionar o gráfico Evolução de Qualidade - Gráfico % de Qualidade ERP, informar o mesmo nome para Nome do Objeto. Utilizando a aba formatar, opção tamanho alterar as dimensões de modo que ocupe todo o espaço disponível na aba.

Na aba % de Qualidade Frente de Caixa, adicionar uma ferramenta do tipo Gráfico, selecionar o gráfico Evolução de Qualidade - Gráfico % de Qualidade Frente de Caixa, informar o mesmo nome para Nome do Objeto. Utilizando a aba formatar, opção tamanho alterar as dimensões de modo que ocupe todo o espaço disponível na aba.

Painel Análise de Defeitos

Na aba Evolução Mensal Backlog de Defeitos, adicionar uma ferramenta do tipo Gráfico, selecionar o gráfico Análise de Defeitos - Evolução Mensal Backlog, informar o

mesmo nome para Nome do Objeto. Utilizando a aba formatar, opção tamanho alterar as dimensões de modo que ocupe todo o espaço disponível na aba.

Na aba Entrada Clientes x Saída Clientes, adicionar uma ferramenta do tipo Gráfico, selecionar o gráfico Análise de Defeitos - Entrada Clientes x Saída Fábrica, informar o mesmo nome para Nome do Objeto. Utilizando a aba formatar, opção tamanho alterar as dimensões de modo que ocupe 80% do espaço disponível na aba, deixando um faixa superior com os 20% de espaço restante.

Ainda na aba Entrada Clientes x Saída Clientes, adicionar cinco ferramentas texto e 5 ferramentas do tipo medidor, especificações de cada ferramenta.

As Figuras 15 e 16, no Capítulo 4, apresentam exemplos de como ficam telas de indicadores de análise de defeitos.

Ferramenta Texto

Nome do Objeto: Texto + Descrição;

Descrição: (% Ponto de Equilibrio, Qtd Ponto de Equilibrio, Média Resolvidos, Média Abertos e Saldo);

Fonte: Tahoma;

Cor de Fundo: No botão, informar os valores conforme padrão RGB (244,244,244);

Tamanho: 12;

Cor do texto: No botão, informar os valores conforme padrão RGB (0,0,0);

Alinhamento horizontal: Centro;

Alinhamento vertical: Topo

Ferramenta Medidor

Apenas selecionar o medidor, informar o mesmo nome em nome do objeto, que são eles: Entrada Clientes x Saída Fábrica - Medidor % Ponto de Equilibrio, Entrada Clientes x Saída Fábrica - Medidor Qtd Ponto de Equilibrio, Entrada Clientes x Saída Fábrica - Medidor Média Resolvidos, Entrada Clientes x Saída Fábrica - Medidor Média Abertos e Entrada Clientes x Saída Fábrica - Medidor Saldo.

Cada texto deve ficar acima de seu respectivo medidor. As ferramentas devem ser dispostas uma ao lado da outra ocupando a faixa de espaço correspondente aos 20% que havia sobrado da aba. A imagem da Figura 16, no Capítulo 4, apresenta como deve ficar o painel e suas abas.

4 RESULTADOS

Este capítulo apresenta o resultado da realização do trabalho.

Na Listagem 1 está o *script* para criação do banco de dados *Data Warehouse* utilizado no processo de BI.

```
db2 "CREATE DATABASE BIFAB AUTOMATIC STORAGE NO ON /db2/dados/DATABASES/BIFAB USING
CODESET UTF-8 TERRITORY BR PAGESIZE 32 K CATALOG TABLESPACE MANAGED BY SYSTEM USING
('/db2/dados/DATABASES/BIFAB/DBPARTITION0/TABLESPACES/DMS/SYSCAT') FILE SYSTEM
CACHING TEMPORARY TABLESPACE MANAGED BY SYSTEM USING
('/db2/dados/DATABASES/BIFAB/DBPARTITION0/TABLESPACES/SMS/SYSTEMP01') FILE SYSTEM
CACHING"

db2 "CREATE TABLESPACE DBA_IDX MANAGED BY DATABASE USING (FILE
'/db2/dados/DATABASES/BIFAB/DBPARTITION0/TABLESPACES/DMS/DBA_IDX.0001' 1G )
AUTORESIZE YES INCREASESIZE 500M MAXSIZE 20G NO FILE SYSTEM CACHING"

db2 "CREATE TABLESPACE DBA_DAT MANAGED BY DATABASE USING (FILE
'/db2/dados/DATABASES/BIFAB/DBPARTITION0/TABLESPACES/DMS/DBA_DAT.0001' 1G )
AUTORESIZE YES INCREASESIZE 500M MAXSIZE 20G NO FILE SYSTEM CACHING"

db2 UPDATE DB CFG FOR BIFAB USING LOGARCHMETH1
"DISK:/db2/dados/DATABASES/BIFAB/LOGS/log_archive" LOGPRIMARY 10 LOGSECOND -1
LOGFILSIZ 10240

db2 UPDATE DB CFG FOR BIFAB USING NEWLOGPATH
/db2/dados/DATABASES/BIFAB/LOGS/log_rotate

db2 BACKUP DB BIFAB  to /dev/null
```

Listagem 1 - *script* para criação do banco de dados Data Warehouse

A Figura 3 apresenta a definição do job Manager. Esse é um exemplo de como um job Manager deve ser criado.

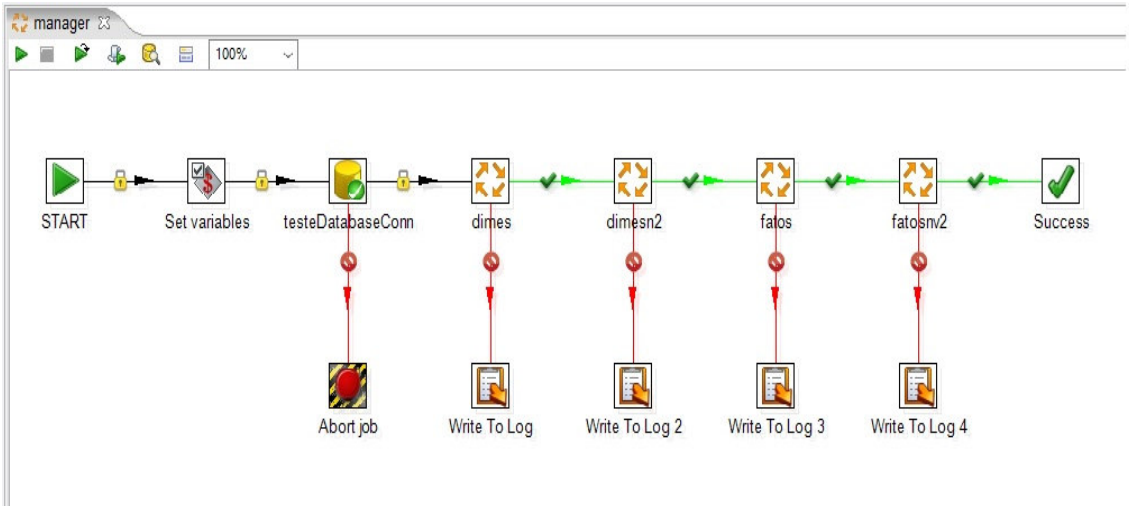

Figura 3 - Job manager criado

Na Figura 4 está a imagem do job Dimes criado.

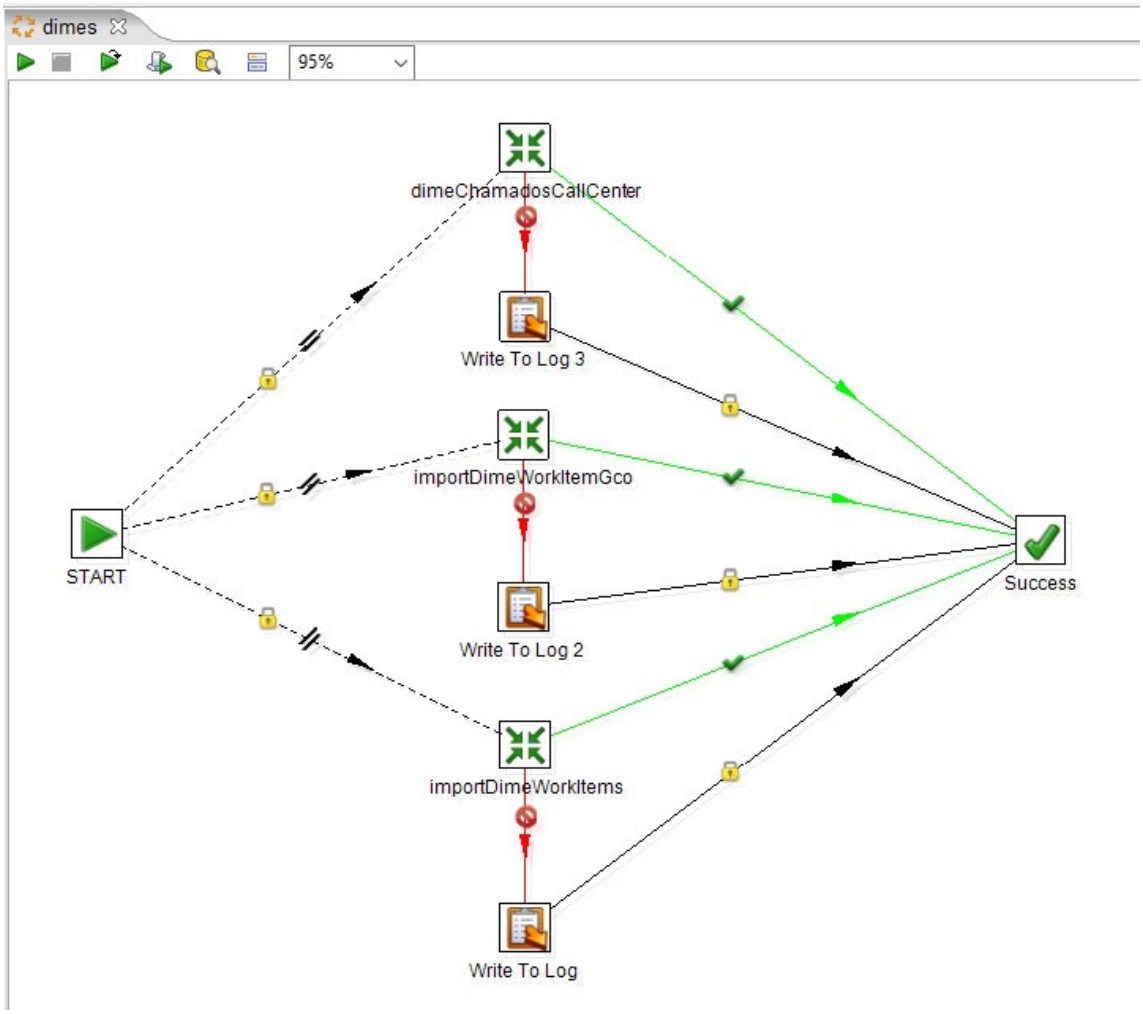

Figura 4 - Job Dimes criado

Um exemplo de como deve ser o transformation DIMECHAMADOSCALLCENTER é apresentado na Figura 5.

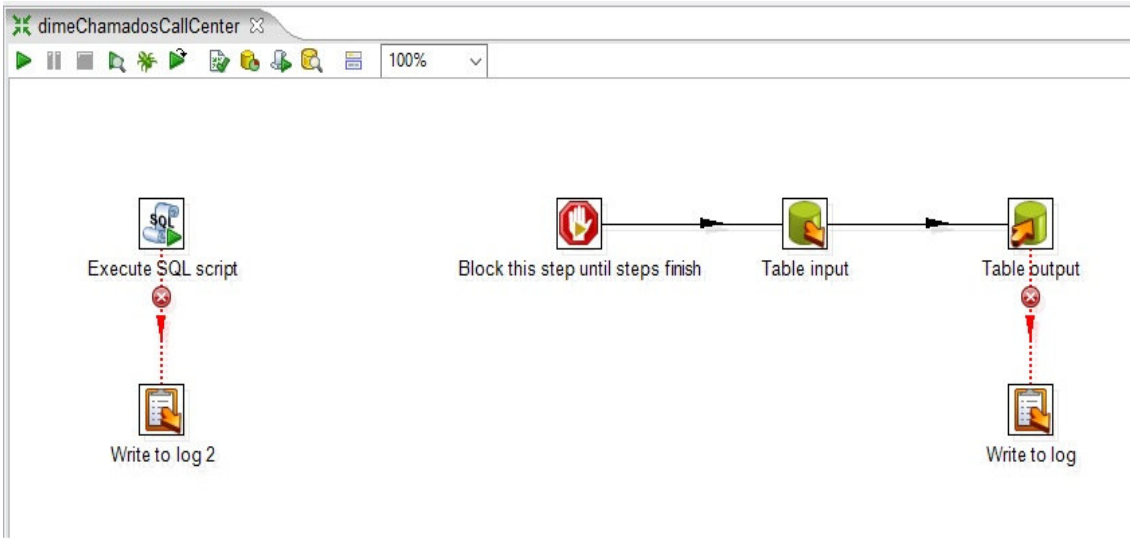

Figura 5 - Transformation dimechamadosCallCenter

A Listagem 2 apresenta um exemplo de como deve ser a instrução SQL do execute SQL script do transformation DIMECHAMADOSCALLCENTER

```
ALTER TABLE DBA.DIME_CHAMADOS_CALLCENTER ACTIVATE NOT LOGGED INITIALLY WITH EMPTY
TABLE;
COMMIT
```

Listagem 2 - Instrução do execute SQL

Um exemplo do table input do transformation DIMECHAMADOSCALLCENTER é apresentado na Listagem 3.

```
WITH                                                                                CH
(IDCLIENTE,IDCHAMADO,IDSITUACAO,IDSISTEMA,IDMODULO,IDITEMMODULO,IDCATEGORIA,DATAHOR
A,FLAGBAIXADA) AS (
SELECT
    CH.IDCLIFOR,
    CH.IDCHAMADO,
    CH.IDSITUACAOCHAMADO,
    CH.IDSISTEMA,
    CH.IDMODULO,
    CH.IDITEMMODULO,
    CH.IDCATEGORIA,
    CH.DATAHORA,
    CH.FLAGBAIXADA
FROM
    DBA.CHAMADOS CH
WHERE
    CH.IDCATEGORIA = 2
)

SELECT
    IDCLIENTE,
    NOMECLIENTE,
    IDCHAMADO,
    SM.IDMODULO,
    SM.DESCRMODULO,
```

```
    SMI.IDITEMMODULO,
    SMI.DESCRITEMMODULO,
    IDCATEGORIA,
    DESCRCATEGORIA,
    IDDIACRIACAO,
    CASE
        WHEN FLAGBAIXADO= 'T' THEN
            IDDIALOG
        ELSE
            NULL
    END AS IDDIARESOLUCAO,
        FLAGBAIXADO,
        IDSITUACAO,
        DESCRSITUACAO,
        IDLISTAESCOPO,
    'Defeito' AS TIPO,
    CASE
        WHEN FLAGBAIXADO = 'T' THEN
            'Baixado'
        ELSE
            NULL
    END AS SITUACAO,
    CASE
        WHEN IDCLIENTE = 885 THEN
            'EMPRESA'
        ELSE
            'CLIENTE'
    END AS ORIGEM
FROM
    (
    SELECT
        CH.IDCLIENTE,
        CF.NOME AS NOMECLIENTE,
        CH.IDCHAMADO,
        CH.IDSISTEMA,
        CH.IDMODULO,
        CH.IDITEMMODULO,
        CC.IDCATEGORIA,
        CC.DESCRCATEGORIA,
        CAST(DATE(CH.DATAHORA) AS INTEGER) AS IDDIACRIACAO,
        CAST(DATE(LOG.DATAHORA) AS INTEGER) AS IDDIALOG,
        ROW_NUMBER() OVER(PARTITION BY CH.IDCHAMADO ORDER BY IDLOG DESC) AS N,
        CH.FLAGBAIXADA AS FLAGBAIXADO,
        CH.IDSITUACAO,
        CS.DESCRSITUACAOCHAMADO AS DESCRSITUACAO,
        A.IDLISTAESCOPO
    FROM
        CH
        JOIN DBA.CLIENTE_FORNECEDOR AS CF ON
            CH.IDCLIENTE = CF.IDCLIFOR
        JOIN DBA.CHAMADOS_LOG AS LOG ON
            CH.IDCHAMADO = LOG.IDCHAMADO
        JOIN DBA.CHAMADOS_CATEGORIA AS CC ON
            CH.IDCATEGORIA = CC.IDCATEGORIA
        JOIN DBA.CHAMADOS_SITUACAO AS CS ON
            CH.IDSITUACAO = CS.IDSITUACAOCHAMADO
        LEFT JOIN VISION.ATIVIDADE AS A ON
            CH.IDCHAMADO = A.IDCHAMADO
    ) AS TMP
    JOIN DBA.SISTEMAS_MODULOS SM ON
        SM.IDMODULO = TMP.IDMODULO AND
        SM.IDSISTEMA = TMP.IDSISTEMA
    JOIN DBA.SISTEMAS_MODULOS_ITEM SMI ON
        SMI.IDITEMMODULO = TMP.IDITEMMODULO AND
        SMI.IDMODULO = TMP.IDMODULO
WHERE
    N = 1
```

Listagem 3 - Table input

A Figura 6 apresenta um exemplo de como deve ser o transformation IMPORTDIMEWORKITEMGCO.

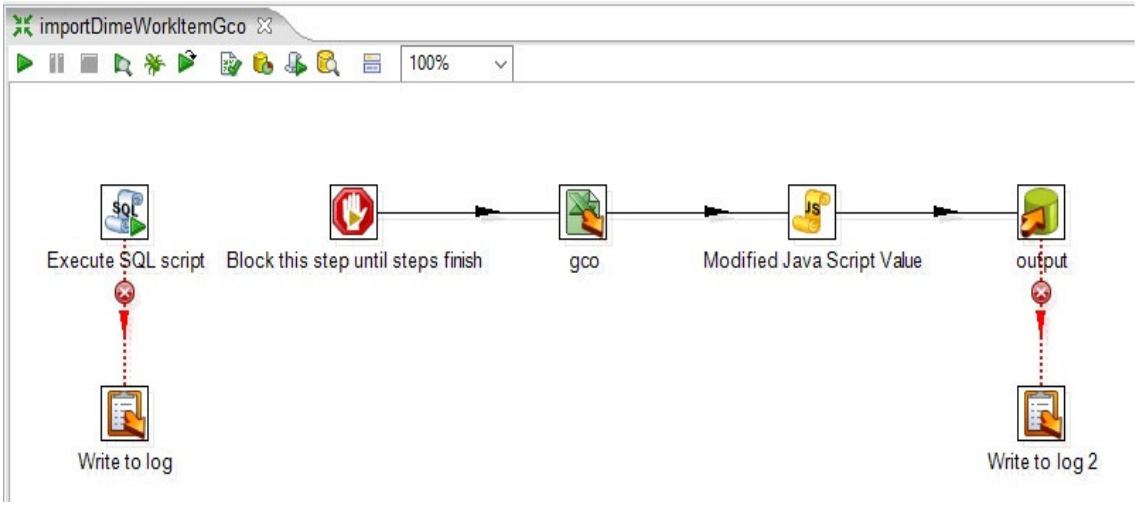

Figura 6 - Transformation importdimeworkitemgco

A Listagem 4 apresenta um exemplo de como deve ser o execute SQL *script* do transformation IMPORTDIMEWORKITEMGCO.

```
ALTER TABLE DBA.DIME_WORK_ITEMS_GCO ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE;
COMMIT
```

Listagem 4 - Execute SQL script do transformation IMPORTDIM

Um exemplo de como deve ser o JavaScript do modified Script values do transformation IMPORTDIMEWORKITEMGCO é apresentado na Listagem 5.

```
var auxDT = null;
var outIDDIACRIACAO = null;

if (Creation_Date != null) {
	auxDT = date2str(Creation_Date,"yyyy-MM-dd");
	outIDDIACRIACAO = auxDT.replace(/[-]+/g,'');
}

var outIDDIARESOLUCAO = null;

if (Resolution_Date != null) {
	auxDT = date2str(Resolution_Date,"yyyy-MM-dd");
	outIDDIARESOLUCAO = auxDT.replace(/[-]+/g,'');
}
```

Listagem 5 - JavaScript do modified values do transformation IMPORTDIMEWORKITEMGCO

A Figura 7 apresenta um exemplo de como deve ser o transformation IMPORTDIMEWORKITEMS.

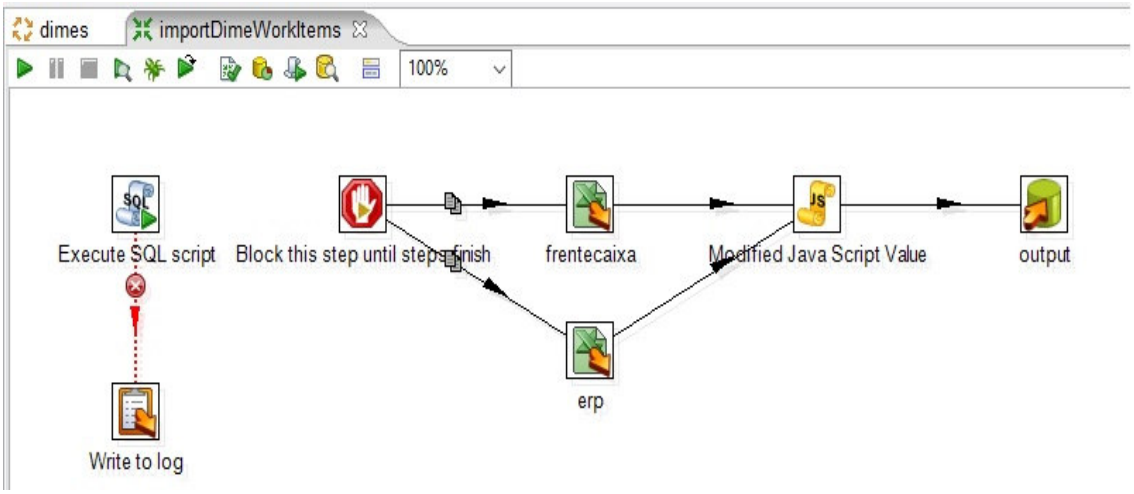

Figura 7 - Transformation IMPORTDIMEWORKITEMS

O execute SQL script do transformation IMPORTDIMEWORKITEMS é exemplificado na Listagem 6.

```
ALTER TABLE DBA.DIME_WORK_ITEMS ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE;
COMMIT
```

Listagem 6 - Execute SQL Script do transformation IMPORTDIMEWORKITEMS

A Listagem 9 apresenta um exemplo de como deve ser o JavaScript do modified values do transformation IMPORTDIMEWORKITEMS

```
var outRELEASESISTEMA = null;
var controle = null;

if (PLANEJADOPARA != null) {

	outRELEASESISTEMA	=	trim(PLANEJADOPARA.replace(/["Build"]|[S][P][0-
9]|[*"()"]|["archived"]+/g,''));
	controle = outRELEASESISTEMA.replace(/["."]+/g,'');

	if(isNaN(controle)){
		outRELEASESISTEMA = null;
	}
}

var outRELEASELIST = null;

if (RELEASELIST != null) {
	outRELEASELIST = RELEASELIST.substring(1);
}

var outPARENT = null;

if (PARENT != null) {
	outPARENT = PARENT.substring(1);
}

var outPLANEJADOPARA = PLANEJADOPARA;

if (outPLANEJADOPARA == null) {
	outPLANEJADOPARA = "Unassigned";
}
```

```
var auxDT = null;
var outIDDIACRIACAO = null;

if (DTCRIACAO != null) {
	auxDT = date2str(DTCRIACAO,"yyyy-MM-dd");
	outIDDIACRIACAO = auxDT.replace(/[-]+/g,'');
}

var outIDDIARESOLUCAO = null;

if (DTRESOLUCAO != null) {
	auxDT = date2str(DTRESOLUCAO,"yyyy-MM-dd");
	outIDDIARESOLUCAO = auxDT.replace(/[-]+/g,'');
}

var outORIGEM = "EMPRESA";

if (CLIENTE == null || CLIENTE.toUpperCase() != "EMPRESA") {
	outORIGEM = "CLIENTE";
}

var outESTIMATIVA = 0;

if (ESTIMATIVA != null) {
	outESTIMATIVA = getTimeInSeconds(ESTIMATIVA)
}

var outTEMPOGASTO = 0;

if (TEMPOGASTO != null) {
	outTEMPOGASTO = getTimeInSeconds(TEMPOGASTO);
}

function getTimeInSeconds(str) {
     var arr = str.split(" ");
     var numero = 0;
     var tempogasto = 0;

    for (i = 0; i < arr.length; i++) {

        if((i % 2) == 0){
            numero = parseFloat(arr[i]);
        }else{
            if( ( arr[i].localeCompare("h") == 0 ) || (arr[i].localeCompare("hour")
== 0 ) || ( arr[i].localeCompare("hours") == 0 ) ){
                tempogasto = tempogasto + (numero*3600);
            }   else   if(   (   arr[i].localeCompare("m")   ==   0   )   ||   (
arr[i].localeCompare("mins") == 0 ) || ( arr[i].localeCompare("min") == 0 ) ){
                tempogasto = tempogasto + (numero*60);
            }   else   if(   (   arr[i].localeCompare("s")   ==   0   )   ||   (
arr[i].localeCompare("secs") == 0 ) || ( arr[i].localeCompare("sec") == 0 ) ){
                tempogasto = tempogasto + numero;
            }   else   if(   (   arr[i].localeCompare("ms")   ==   0   )   ||   (
arr[i].localeCompare("millisecs") == 0 ) || ( arr[i].localeCompare("millisec") == 0
) ){
                tempogasto = tempogasto + (numero/1000);
            }
        }
    }

    return tempogasto;
}
```

Listagem 7 - JavaScript do modified JavaScript values do transformation IMPORTDIME

O JOB DIMESNV2 é exemplificado na Figura 8.

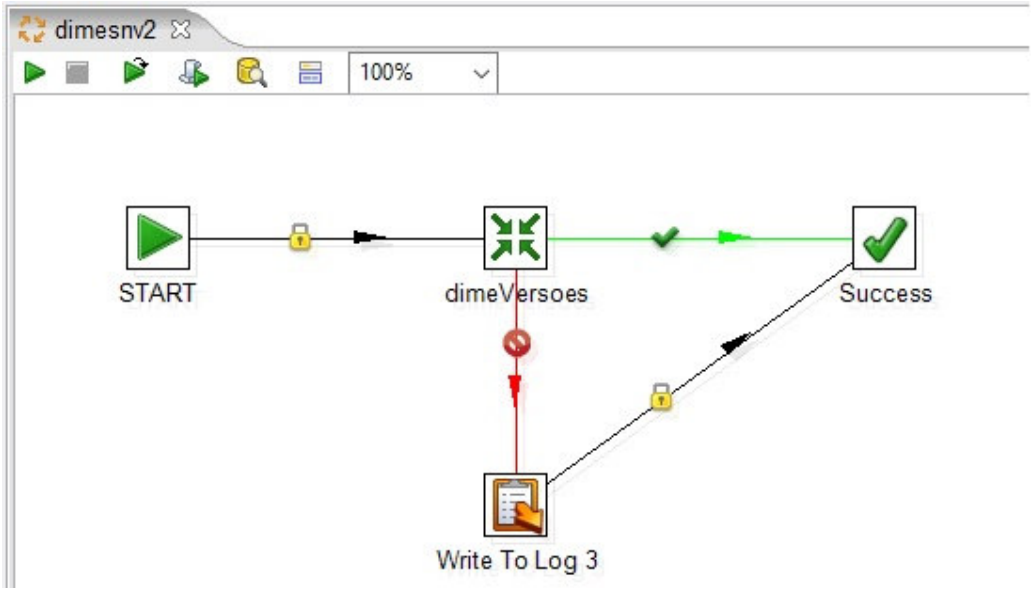

Figura 8 - Job DIMESNV2

Um exemplo de como deve ser o transformation DIMEVERSOES é apresentado na Figura 9.

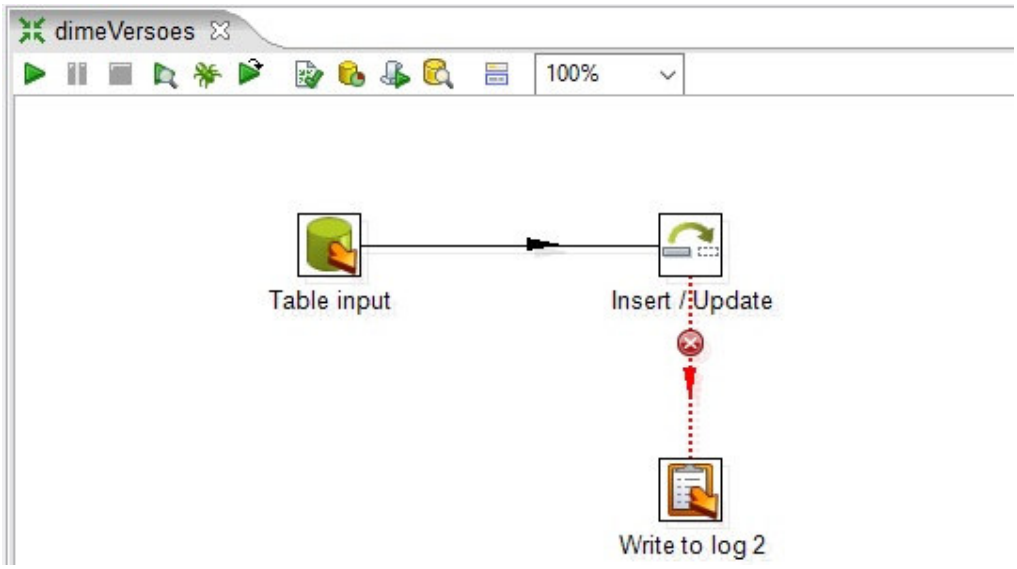

Figura 9 - Transformation DIMEVERSOES

A Listagem 8 apresenta um exemplo de como deve ser o table input do transformation DIMEVERSOES.

```
WITH LOOPQUERY (WI_ID, WORDNUM, WORD, RESTANTE) AS (
SELECT
    WI_ID,
    1,
    CASE
        WHEN LOCATE('#', RELEASELIST) > 0 THEN
            SUBSTR (RELEASELIST, 1, LOCATE ('#', RELEASELIST) - 1)
```

```
        ELSE
            RELEASELIST
    END WORD,
    CASE
        WHEN LOCATE ('#', RELEASELIST) > 0 THEN
            LTRIM (SUBSTR (RELEASELIST, LOCATE ('#', RELEASELIST) + 1))
        ELSE
            NULL
    END RESTANTE
FROM
    DBA.DIME_WORK_ITEMS BASE

UNION ALL

SELECT
    WI_ID,
    WORDNUM + 1,
    CASE
        WHEN LOCATE ('#', RESTANTE) > 0 THEN
            SUBSTR (RESTANTE, 1, LOCATE ('#', RESTANTE) - 1)
        ELSE
            RESTANTE
    END WORD,
    CASE
        WHEN LOCATE ('#', RESTANTE) > 0 THEN
            LTRIM (SUBSTR (RESTANTE, LOCATE ('#', RESTANTE) + 1))
    ELSE
        NULL
    END RESTANTE
FROM
    LOOPQUERY LQ
WHERE
    LQ.RESTANTE IS NOT NULL
)

SELECT
    IDVERSAO,
    DESCRVERSAO,
    CASE WHEN RIGHT(IDVERSAO,1) > 0 THEN
        'BUILD'
    ELSE
        'RELEASE'
    END AS TIPO
FROM
    (
    SELECT
        DBA.DESCRRELEASETOINT(REPLACE(REPLACE(DESCRVERSAO,CHR(13),''),CHR(10),''))
AS IDVERSAO,
        REPLACE(REPLACE(DESCRVERSAO,CHR(13),''),CHR(10),'') AS DESCRVERSAO
    FROM
        (
        SELECT
            WORD AS DESCRVERSAO
        FROM
            LOOPQUERY LQ
        WHERE
            (WORD IS NOT NULL OR
            WORD <> '')
        ) AS TMP
    WHERE
        LENGTH(DESCRVERSAO) >= 7 AND

DBA.ISNUMBER(REPLACE(REPLACE(REPLACE(DESCRVERSAO,CHR(13),''),CHR(10),''),'.',''))  =
1
    ) AS TMP3
GROUP BY
    IDVERSAO,
    DESCRVERSAO
```

```
UNION

SELECT
    IDVERSAO,
    DESCRVERSAO,
    CASE WHEN RIGHT(IDVERSAO,1) > 0 THEN
        'BUILD'
    ELSE
        'RELEASE'
    END AS TIPO
FROM
    (
    SELECT
        DBA.DESCRRELEASETOINT(REPLACE(REPLACE(LOCALIZADOEM,CHR(13),''),CHR(10),''))
AS IDVERSAO,
        LOCALIZADOEM AS DESCRVERSAO
    FROM
        DBA.DIME_WORK_ITEMS WI
    WHERE
        LOCALIZADOEM IS NOT NULL AND
        LENGTH(LOCALIZADOEM) >= 7 AND

DBA.ISNUMBER(REPLACE(REPLACE(REPLACE(LOCALIZADOEM,CHR(13),''),CHR(10),''),'.',''))
= 1
    GROUP BY
        LOCALIZADOEM

    UNION

    SELECT

DBA.DESCRRELEASETOINT(REPLACE(REPLACE(RELEASESISTEMA,CHR(13),''),CHR(10),''))     AS
IDVERSAO,
        RELEASESISTEMA AS DESCRVERSAO
    FROM
        DBA.DIME_WORK_ITEMS WI
    WHERE
        RELEASESISTEMA IS NOT NULL AND
        LENGTH(RELEASESISTEMA) >= 7
    GROUP BY
        RELEASESISTEMA
    ) AS TMP
```

Listagem 8 - Table input do transformation DIMEVERSOES

Um exemplo de como deve ser o job FATOS é apresentado na Figura 10.

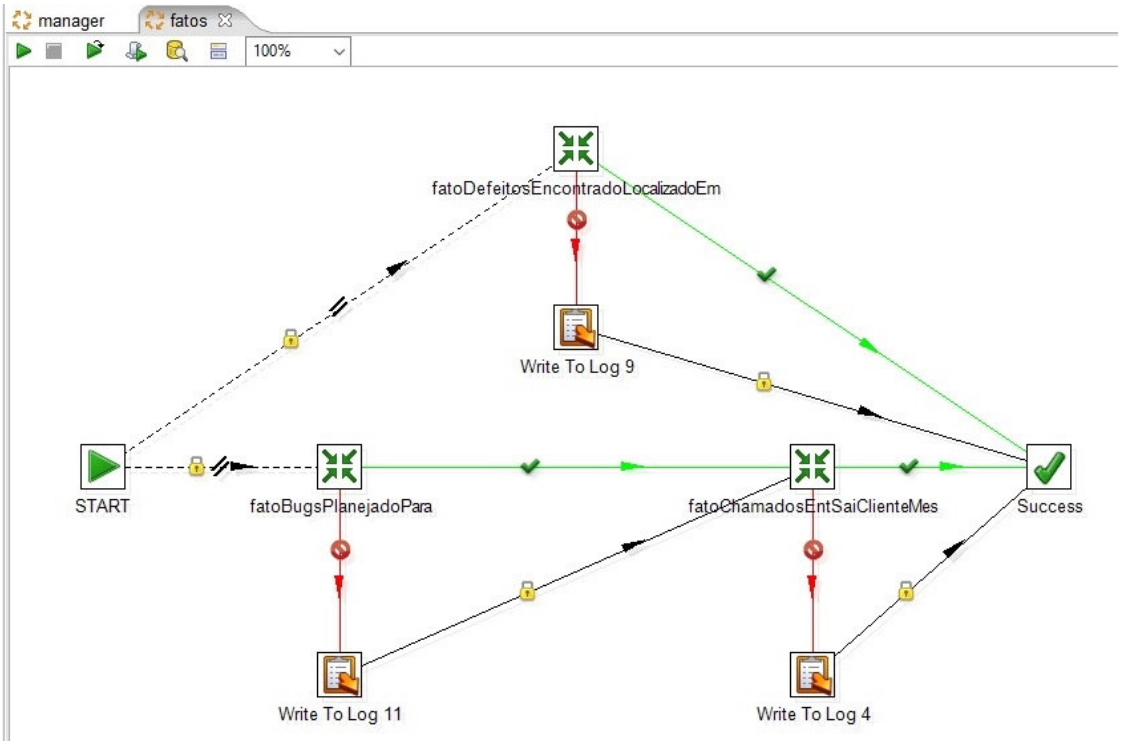

Figura 10 - Job FATOS

Um exemplo de como deve ser o transformation FATODEFEITOSENCONTRADOLOCALIZADOEM é apresentado na Figura 11.

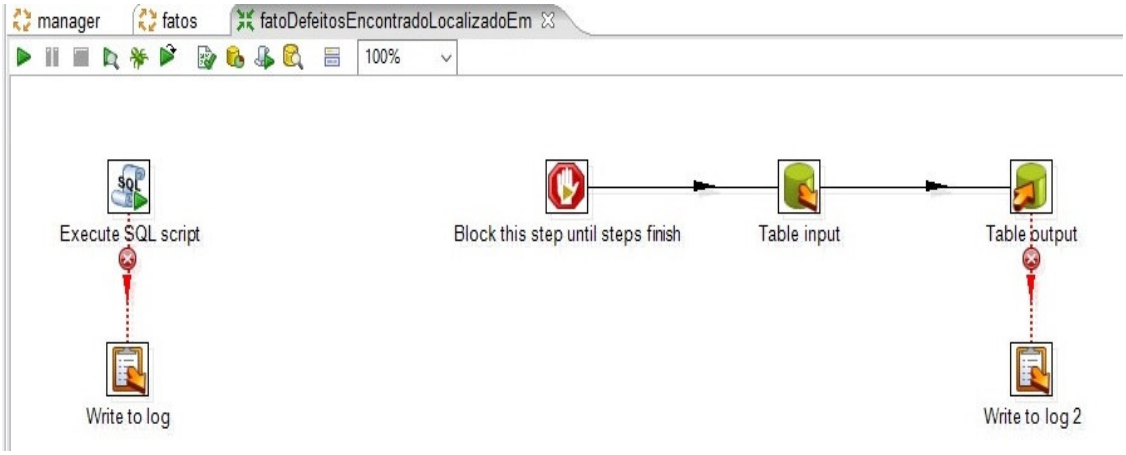

Figura 11 - Transformation FATODEFEITOSENCONTRADOLOCALIZADOEM

Na Listagem 9 está um exemplo de como deve ser o execute SQL script do transformation FATODEFEITOSENCONTRADOLOCALIZADOEM.

```
ALTER TABLE DBA.FATO_DEFEITOS_ENCONTRADO_LOCALIZADO_EM ACTIVATE NOT LOGGED
INITIALLY WITH EMPTY TABLE;
COMMIT
```

Listagem 9 - Execute SQL script do transformation FATODEFEITOSENCONTRADOLOCALIZADOE

A Listagem 10 apresenta um exemplo de como deve ser o table input do transformation FATODEFEITOSENCONTRADOLOCALIZADOEM.

```
SELECT
    AREADEPROJETO,
    ARQUIVADOEM,
    IDVERSAO,
    SUM(DEFEITOEMPRODUCAO) AS DEFEITOSEMPRODUCAO,
    SUM(DEFEITOEMHOMOLOGACAO) AS DEFEITOSEMHOMOLOGACAO
FROM
    (
    SELECT
        AREADEPROJETO,
        ARQUIVADOEM,
        CASE
            WHEN RIGHT(IDVERSAO,1) > 0 THEN
                COALESCE(DBA.IDVERSION_OF_BUILD(IDVERSAO),IDVERSAO)
            ELSE
                IDVERSAO
        END AS IDVERSAO,
        DEFEITOEMPRODUCAO,
        DEFEITOEMHOMOLOGACAO
    FROM
        (
        SELECT
            AREADEPROJETO,
            COALESCE(ARQUIVADOEM,'Unassigned') AS ARQUIVADOEM,

DBA.DESCRRELEASETOINT(REPLACE(REPLACE(LOCALIZADOEM,CHR(13),''),CHR(10),''))       AS
IDVERSAO,
            CASE
                WHEN ENCONTRADOEMPRODUCAO = 'true' THEN
                    1
                ELSE
                    0
            END AS DEFEITOEMPRODUCAO,
            CASE
                WHEN ENCONTRADOEMPRODUCAO = 'false' THEN
                    1
                ELSE
                    0
            END AS DEFEITOEMHOMOLOGACAO
        FROM
            DBA.DIME_WORK_ITEMS WI
        WHERE
            LOCALIZADOEM IS NOT NULL AND
            LENGTH(LOCALIZADOEM) >= 7 AND
            WI.TIPO = 'Defeito' AND
            WI.STATUS <> 'Cancelado' AND

DBA.ISNUMBER(REPLACE(REPLACE(REPLACE(LOCALIZADOEM,CHR(13),''),CHR(10),''),'.',''))
= 1
        ) AS TMP
    ) AS TMP2
GROUP BY
    AREADEPROJETO,
    ARQUIVADOEM,
    IDVERSAO
```

Listagem 10 - Table input do transformation FATODEFEITOSENCONTRADOLOCALIZADOEM

A Figura 12 apresenta um exemplo de como deve ser o transformation FATOBUGSPLANEJADOPARA.

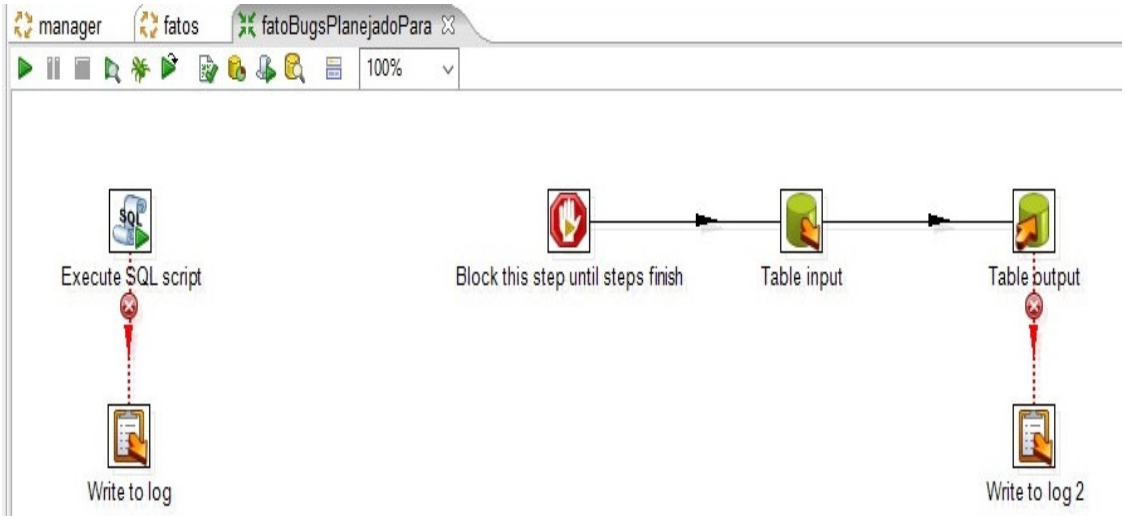

Figura 12 - Transformation FATOBUGSPLANEJADOPARA

A Listagem 11 apresenta um exemplo de como deve ser o execute SQL script do transformation FATOBUGSPLANEJADOPARA.

```
ALTER TABLE DBA.FATO_BUGS_PLANEJADO_PARA ACTIVATE NOT LOGGED INITIALLY WITH EMPTY
TABLE;
COMMIT
```

Listagem 11 - Execute SQL script do transformation FATOBUGSPLANEJADOPARA

Um exemplo de como deve ser o table input do transformation FATOBUGSPLANEJADOPARA é apresentado na Listagem 12.

```
SELECT
    AREADEPROJETO,
    ARQUIVADOEM,
    IDVERSAO,
    SUM(BUG) AS BUGS
FROM
    (
    SELECT
        AREADEPROJETO,
        ARQUIVADOEM,
        CASE
            WHEN RIGHT(IDVERSAO,1) > 0 THEN
                COALESCE(DBA.IDVERSION_OF_BUILD(IDVERSAO),IDVERSAO)
            ELSE
                IDVERSAO
        END AS IDVERSAO,
        BUG
    FROM
        (
        SELECT
            AREADEPROJETO,
            COALESCE(ARQUIVADOEM,'Unassigned') AS ARQUIVADOEM,

DBA.DESCRRELEASETOINT(REPLACE(REPLACE(RELEASESISTEMA,CHR(13),''),CHR(10),''))     AS
IDVERSAO,
            1 AS BUG
        FROM
            DBA.DIME_WORK_ITEMS WI
        WHERE
            RELEASESISTEMA IS NOT NULL AND
            LENGTH(RELEASESISTEMA) >= 7 AND
```

```
            (WI.TIPO = 'Bug' AND WI.STATUS IN ('Pronto','Resolvido'))
        ) AS TMP
    ) AS TMP2
GROUP BY
    AREADEPROJETO,
    ARQUIVADOEM,
    IDVERSAO
```

Listagem 12 - Table input do transformation FATOBUGSPLANEJADOPARA

Na Figura 13 está um exemplo de como deve ser o transformation FATOCHAMADOSENTSAICLIENTEMES.

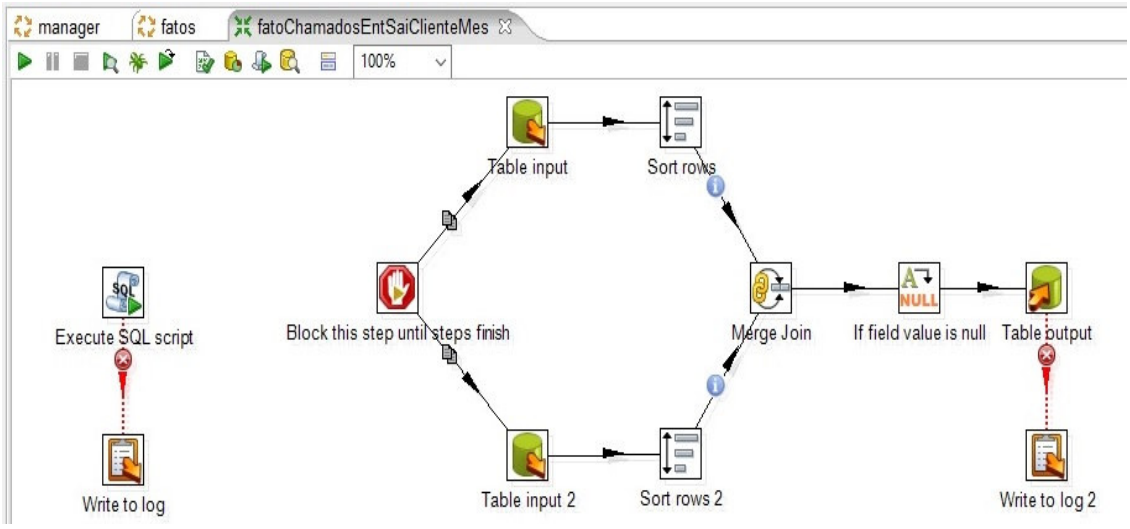

Figura 13 - Transformation FATOSALDOBACKLOGMES

O execute SQL script do transformation FATOCHAMADOSENTSAICLIENTEMES é apresentado na Listagem 13.

```
ALTER TABLE DBA.FATO_CHAMADOS_ENTSAI_CLIENTE_MES ACTIVATE NOT LOGGED INITIALLY WITH
EMPTY TABLE;
COMMIT
```

Listagem 13 - Execute SQL script do transformation FATOCHAMADOSENTSAICLIENTEMES

A Listagem 14 apresenta um exemplo de como deve ser o table input do transformation FATOCHAMADOSENTSAICLIENTEMES.

```
SELECT
    COUNT(TMP1.TICKETID) AS QTDDEFEITOSABERTOS,
    YEAR(TMP1.OWNDATE) || RIGHT('0'|| MONTH(TMP1.OWNDATE),2) AS IDMES
FROM
    (
    SELECT
        TK.TICKETID,
        MAX(TK.OWNDATE) AS OWNDATE
    FROM
        MAXIMO.TKOWNERHISTORY AS TK
        LEFT OUTER JOIN MAXIMO.SR AS SR ON SR.TICKETID = TK.TICKETID
    WHERE
        (TK.OWNER    IN    ('GESTAOMUDANCAS','GMUDANCA')    OR    TK.OWNERGROUP    IN
('GMUDANCA')) AND
        SR.EMPRESA_TIPODEMANDA IN ('Erro Operacional', 'Erro de Sistema', 'Prob. de
Infraestrutura') AND
```

```
            SR.PLUSPCUSTOMER <> 885
    GROUP BY
        TK.TICKETID
    )AS TMP1
GROUP BY
    YEAR(TMP1.OWNDATE) || RIGHT('0'|| MONTH(TMP1.OWNDATE),2)
```

Listagem 14 - Table input do transformation FATOCHAMADOSENTSAICLIENTEMES

Na Listagem 15 está um exemplo de como deve ser o table input 2 do transformation FATOCHAMADOSENTSAICLIENTEMES.

```
SELECT
    SUM(QTDRESOLVIDOS) AS QTDDEFEITOSRESOLVIDOS,
    IDMES
FROM
    (
    SELECT
        COUNT(1) AS QTDRESOLVIDOS,
        ANOMES AS IDMES
    FROM
        DBA.DIME_WORK_ITEMS STG_WORK_ITEMS
        JOIN DBA.DIME_TEMPO_DIA DIME_TEMPO_DIA ON
                    DIME_TEMPO_DIA.DIA = STG_WORK_ITEMS.DTRESOLUCAO
    WHERE
        TIPO = 'Defeito' AND
        (CLIENTE IS NULL OR UPPER(CLIENTE) NOT LIKE '%EMPRESA%') AND
             STG_WORK_ITEMS.STATUS <> 'Cancelado'
    GROUP BY
        ANOMES
    ) AS TMP
GROUP BY
    IDMES
```

Listagem 15 - Table input 2 do transformation FATOCHAMADOSENTSAICLIENTEMES

Na Figura 14 está um exemplo de como deve ser o job FATOSNV2.

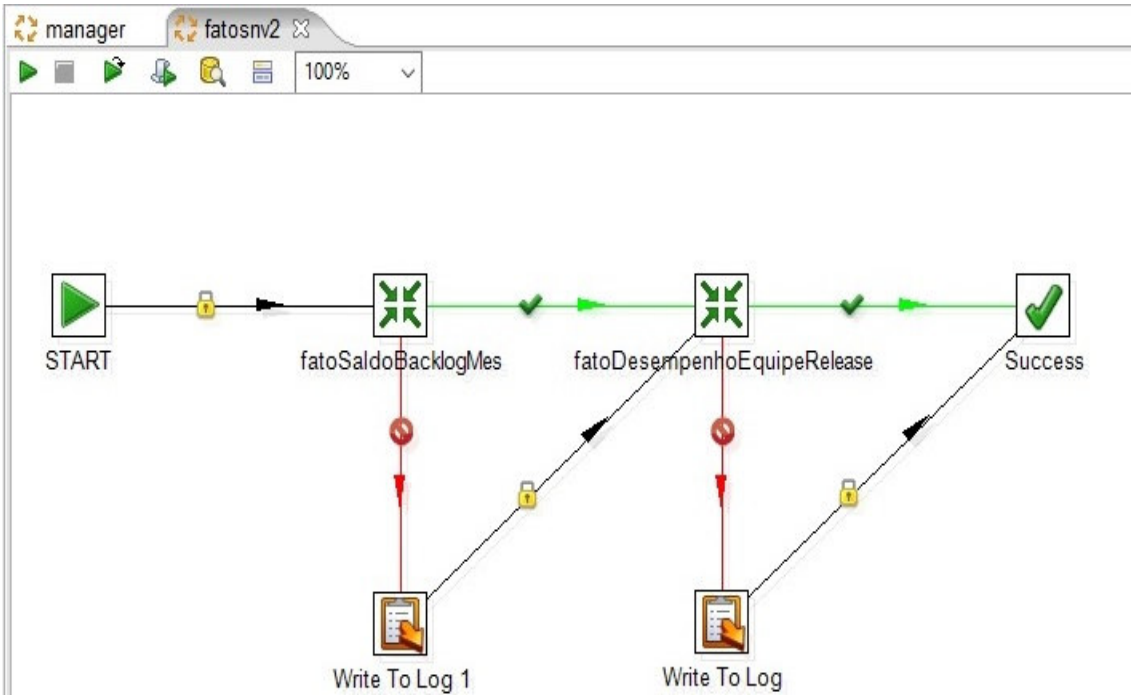

Figura 14 - Job FATOSNV2

Na Figura 15 está um exemplo de como deve ser o transformation FATOSALDOBACKLOGMES.

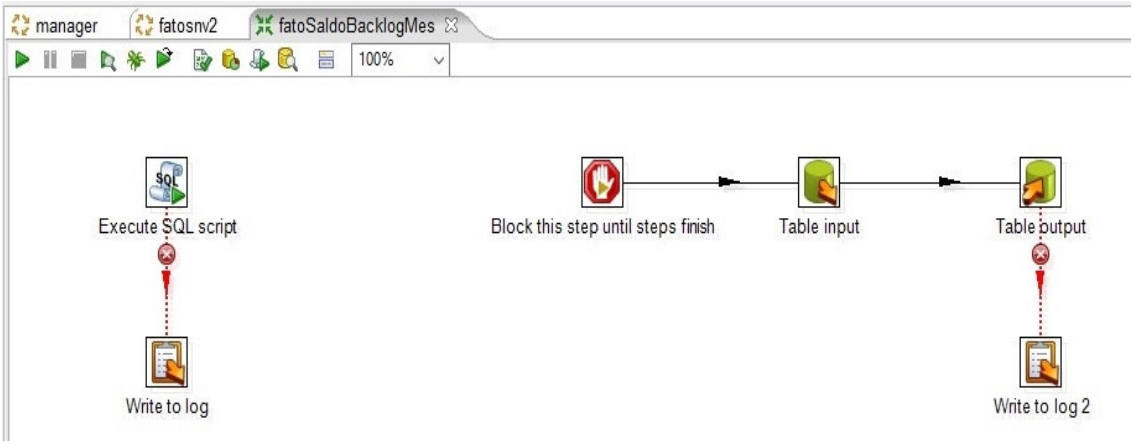

Figura 15 - Transformation FATOSALDOBACKLOGMES

A Listagem 16 apresenta um exemplo de como deve ser o execute sql script do transformation FATOSALDOBACKLOGMES.

```
ALTER TABLE DBA.FATO_SALDO_BACKLOG_MES ACTIVATE NOT LOGGED INITIALLY WITH EMPTY
TABLE;
COMMIT
```

Listagem 16 - Execute SQL script do transformation FATOSALDOBACKLOGMES

Um exemplo de como deve ser o table input do transformation FATOSALDOBACKLOGMES é apresentado na Listagem 17.

```
WITH TABELA_DATA (DIA, NUMEROANO, NUMEROMES, ABRMES, ANOMES) AS (
SELECT
    DIME_TEMPO_MES.DATAULTIMODIAMES,
    DIAS.NUMEROANO,
    DIAS.NUMEROMES,
    DIAS.ABRMES,
    DIAS.ANOMES
FROM
    DBA.DIME_TEMPO_MES DIME_TEMPO_MES,
    (
    SELECT
        NUMEROMES,
        NUMEROANO,
        ABRMES,
        ANOMES
    FROM
        DBA.DIME_TEMPO_DIA DIME_TEMPO_DIA
    WHERE
        DIA >= (SELECT MIN(DTCRIACAO) FROM DBA.DIME_WORK_ITEMS WHERE TIPO =
'Defeito') AND
        DIA <= CURRENT DATE
    GROUP BY
        NUMEROMES,
        NUMEROANO,
        ABRMES,
        ANOMES
    ) AS DIAS
WHERE
```

```
    DIME_TEMPO_MES.NUMEROMES = DIAS.NUMEROMES AND
    DIME_TEMPO_MES.NUMEROANO = DIAS.NUMEROANO
)

SELECT
    SUM(ORIGEMCLIENTECALLCENTER) AS ORIGEMCLIENTECALLCENTER,
    SUM(ORIGEMEMPRESACALLCENTER) AS ORIGEMEMPRESACALLCENTER,
    SUM(ORIGEMCLIENTEJAZZ) AS ORIGEMCLIENTEJAZZ,
    SUM(ORIGEMEMPRESAJAZZ) AS ORIGEMEMPRESAJAZZ,
    IDMES
FROM
    (
    SELECT
        CASE
            WHEN ORIGEMCLIENTE < 0 THEN
                ORIGEMCLIENTE * -1
            ELSE
                ORIGEMCLIENTE
        END AS ORIGEMCLIENTECALLCENTER,
        CASE
            WHEN ORIGEMEMPRESA < 0 THEN
                ORIGEMEMPRESA * -1
            ELSE
                ORIGEMEMPRESA
        END AS ORIGEMEMPRESACALLCENTER,
        0 AS ORIGEMCLIENTEJAZZ,
        0 AS ORIGEMEMPRESAJAZZ,
        IDMES
    FROM
        (
        SELECT
            SUM(ORIGEMCLIENTE) AS ORIGEMCLIENTE,
            SUM(ORIGEMEMPRESA) AS ORIGEMEMPRESA,
            IDMES
        FROM
            (
            SELECT
                CASE
                    WHEN ORIGEM = 'CLIENTE' THEN
                        COUNT(1)
                    ELSE
                        0
                END AS ORIGEMCLIENTE,
                CASE
                    WHEN ORIGEM = 'EMPRESA' THEN
                        COUNT(1)
                    ELSE
                        0
                END AS ORIGEMEMPRESA,
                ANOMES AS IDMES
            FROM
                DBA.DIME_CHAMADOS_CALLCENTER AS SCC,
                TABELA_DATA AS DT
            WHERE
                SCC.DTCRIACAO <= DT.DIA AND
                SCC.TIPO = 'Defeito'
            GROUP BY
                ANOMES,
                ORIGEM

            UNION ALL

            SELECT
                CASE
                    WHEN ORIGEM = 'CLIENTE' THEN
                        COUNT(1) * -1
                    ELSE
                        0
```

```
            END AS QTDORIGEMCLIENTE,
            CASE
                WHEN ORIGEM = 'EMPRESA' THEN
                    COUNT(1) * -1
                ELSE
                    0
            END AS QTDORIGEMEMPRESA,
            ANOMES AS IDMES
        FROM
            DBA.DIME_CHAMADOS_CALLCENTER AS SCC,
            TABELA_DATA AS DT
        WHERE
            SCC.DTRESOLUCAO <= DT.DIA AND
            SCC.SITUACAO = 'Baixado' AND
            SCC.TIPO = 'Defeito'
        GROUP BY
            ANOMES,
            ORIGEM
        ) AS TMP
    GROUP BY
        IDMES
    ) AS TMP

UNION ALL

SELECT
    0 AS ORIGEMCLIENTECALLCENTER,
    0 AS ORIGEMEMPRESACALLCENTER,
    CASE
        WHEN ORIGEMCLIENTE < 0 THEN
            ORIGEMCLIENTE * -1
        ELSE
            ORIGEMCLIENTE
    END AS ORIGEMCLIENTEJAZZ,
    CASE
        WHEN ORIGEMEMPRESA < 0 THEN
            ORIGEMEMPRESA * -1
        ELSE
            ORIGEMEMPRESA
    END AS ORIGEMEMPRESAJAZZ,
    IDMES
FROM
    (
    SELECT
        SUM(ORIGEMCLIENTE) AS ORIGEMCLIENTE,
        SUM(ORIGEMEMPRESA) AS ORIGEMEMPRESA,
        IDMES
    FROM
        (
        SELECT
            CASE
                WHEN ORIGEM = 'CLIENTE' THEN
                    COUNT(1)
                ELSE
                    0
            END AS ORIGEMCLIENTE,
            CASE
                WHEN ORIGEM = 'EMPRESA' THEN
                    COUNT(1)
                ELSE
                    0
            END AS ORIGEMEMPRESA,
            ANOMES AS IDMES
        FROM
            DBA.DIME_WORK_ITEMS AS WI,
            TABELA_DATA AS DT
        WHERE
            WI.DTCRIACAO <= DT.DIA AND
```

```
                TIPO = 'Defeito'
            GROUP BY
                ANOMES,
                ORIGEM

            UNION ALL

            SELECT
                CASE
                    WHEN ORIGEM = 'CLIENTE' THEN
                        COUNT(1) * -1
                    ELSE
                        0
                END AS QTDORIGEMCLIENTE,
                CASE
                    WHEN ORIGEM = 'EMPRESA' THEN
                        COUNT(1) * -1
                    ELSE
                        0
                END AS QTDORIGEMEMPRESA,
                ANOMES AS IDMES
            FROM
                DBA.DIME_WORK_ITEMS AS WI,
                TABELA_DATA AS DT
            WHERE
                WI.DTRESOLUCAO <= DT.DIA AND
                WI.DTRESOLUCAO IS NOT NULL AND
                TIPO = 'Defeito'
            GROUP BY
                ANOMES,
                ORIGEM
            ) AS TMP
        GROUP BY
            IDMES
        ) AS TMP
    ) AS TB
GROUP BY
    IDMES
```

Listagem 17 - Table input do transformation FATOSALDOBACKLOGMES

A Figura 16 apresenta um exemplo de como deve ser o transformation FATODESEMPENHOEQUIPERELEASE

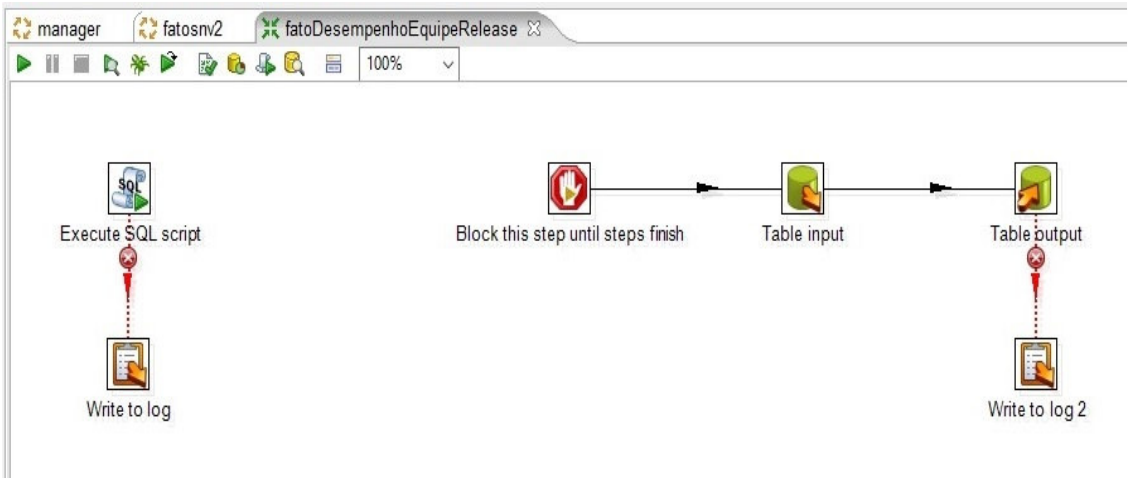

Figura 16 - Transformation FATODESEMPENHOEQUIPERELEASEFATOSALDOBACKLOGMES

A instrução SQL utilizada no execute SQL script do transformation FATODESEMPENHOEQUIPERELEASE é apresentada na Listagem 18.

```
ALTER TABLE DBA.FATO_DESEMPENHO_EQUIPE_RELEASE ACTIVATE NOT LOGGED INITIALLY WITH
EMPTY TABLE;
COMMIT
```

Listagem 18 - Execute SQL script do transformation FATODESEMPENHOEQUIPERELEASE

A instrução SQL do tipo select utilizado no table input do transformation FATODESEMPENHOEQUIPERELEASE está na Listagem 19.

```
SELECT
    AREADEPROJETO,
    ARQUIVADOEM,
    IDVERSAO,
    QTDAUDITORIA,
    QTDBUG,
    ESCOPO_ENTREGUE,
    QTDPRIORIDADE1,
    QTDDEFEITOEMPRODUCAO,
    QTDDEFEITOEMHOMOLOGACAO,
    CASE
        WHEN QTDDEFEITOEMHOMOLOGACAO + QTDDEFEITOEMPRODUCAO + QTDBUG = 0 THEN
            0
        ELSE
            CAST( 100 * ( 1 - ( QTDDEFEITOEMPRODUCAO / ( ( QTDDEFEITOEMHOMOLOGACAO
+ QTDDEFEITOEMPRODUCAO + QTDBUG ) * 1.00 ) ) ) AS DECIMAL(15,2))
    END AS PERQUALIDADE
FROM
    (
    SELECT
        AREADEPROJETO,
        ARQUIVADOEM,
        IDVERSAO,
        SUM(QTDAUDITORIA) AS QTDAUDITORIA,
        SUM(ESCOPO_ENTREGUE) AS ESCOPO_ENTREGUE,
        SUM(QTDPRIORIDADE1) AS QTDPRIORIDADE1,
        SUM(QTDBUG) AS QTDBUG,
        SUM(QTDDEFEITOEMPRODUCAO) AS QTDDEFEITOEMPRODUCAO,
        SUM(QTDDEFEITOEMHOMOLOGACAO) AS QTDDEFEITOEMHOMOLOGACAO
    FROM
        (
        SELECT
            AREADEPROJETO,
            ARQUIVADOEM,
            IDVERSAO,
            SUM(AUDITORIA) AS QTDAUDITORIA,
            SUM(HISTORIA) + SUM(DEFEITO) AS ESCOPO_ENTREGUE,
            SUM(PRIORIDADE1) AS QTDPRIORIDADE1,
            0 AS QTDBUG,
            0 AS QTDDEFEITOEMPRODUCAO,
            0 AS QTDDEFEITOEMHOMOLOGACAO
        FROM
            (
            SELECT
                AREADEPROJETO,
                ARQUIVADOEM,
                CASE
                    WHEN RIGHT(IDVERSAO,1) > 0 THEN
                        COALESCE(DBA.IDVERSION_OF_BUILD(IDVERSAO),IDVERSAO)
                    ELSE
                        IDVERSAO
                END AS IDVERSAO,
                AUDITORIA,
```

```
                HISTORIA,
                DEFEITO,
                PRIORIDADE1,
                0 AS QTDBUG,
                0 AS QTDDEFEITOEMPRODUCAO,
                0 AS QTDDEFEITOEMHOMOLOGACAO
            FROM
                (
                SELECT
                    AREADEPROJETO,
                    ARQUIVADOEM,

DBA.DESCRRELEASETOINT(REPLACE(REPLACE(RELEASESISTEMA,CHR(13),''),CHR(10),''))     AS
IDVERSAO,
                    CASE
                        WHEN
                            WI_GCO.TIPO = 'Auditoria' AND WI.TIPO = 'Defeito' THEN
                                1
                        ELSE
                            0
                    END AS AUDITORIA,
                    CASE
                        WHEN  WI.TIPO  =  'Defeito'  AND  WI.PRIORIDADE  <>  'SM'  AND
(WI_GCO.TIPO <> 'Auditoria' OR WI_GCO.TIPO IS NULL) THEN
                            1
                        ELSE
                            0
                    END AS DEFEITO,
                    CASE
                        WHEN  WI.TIPO  =  'Defeito'  AND  WI.PRIORIDADE  =  'SM'  AND
(WI_GCO.TIPO <> 'Auditoria' OR WI_GCO.TIPO IS NULL) THEN
                            1
                        ELSE
                            0
                    END AS PRIORIDADE1,
                    CASE
                        WHEN WI.TIPO IN ('Historia','Story') THEN
                            1
                        ELSE
                            0
                    END AS HISTORIA
                FROM
                    DBA.DIME_WORK_ITEMS WI
                    LEFT JOIN DBA.DIME_WORK_ITEMS_GCO WI_GCO ON
                        WI.PARENT = WI_GCO.WI_ID
                WHERE
                    RELEASESISTEMA IS NOT NULL AND
                    LENGTH(RELEASESISTEMA) >= 7 AND
                    ((WI.TIPO      =      'Defeito'      AND      WI.STATUS      IN
('Aceito','Documentado','Homologando','Pronto')) OR
                    (WI.TIPO    IN    ('Historia','Story')    AND    WI.STATUS    IN
('Aceito','Documentado','Homologando','Pronto')))
                ) AS TMP
            ) AS TMP2
        GROUP BY
            AREADEPROJETO,
            ARQUIVADOEM,
            IDVERSAO

        UNION ALL

        SELECT
            AREADEPROJETO,
            ARQUIVADOEM,
            IDVERSAO,
            0 AS QTDAUDITORIA,
            0 AS ESCOPO_ENTREGUE,
            0 AS QTDPRIORIDADE1,
```

```
            0 AS QTDBUG,
            DEFEITOSEMPRODUCAO AS QTDDEFEITOEMPRODUCAO,
            DEFEITOSEMHOMOLOGACAO AS QTDDEFEITOEMHOMOLOGACAO
        FROM
            DBA.FATO_DEFEITOS_ENCONTRADO_LOCALIZADO_EM
FATO_DEFEITOS_ENCONTRADO_LOCALIZADO_EM

        UNION ALL

        SELECT
            AREADEPROJETO,
            ARQUIVADOEM,
            IDVERSAO,
            0 AS QTDAUDITORIA,
            0 AS ESCOPO_ENTREGUE,
            0 AS QTDPRIORIDADE1,
            BUGS AS QTDBUG,
            0 AS QTDDEFEITOEMPRODUCAO,
            0 AS QTDDEFEITOEMHOMOLOGACAO
        FROM
            DBA.FATO_BUGS_PLANEJADO_PARA FATO_BUGS_PLANEJADO_PARA
        ) AS TMP3
    GROUP BY
        AREADEPROJETO,
        ARQUIVADOEM,
        IDVERSAO
    ) AS TMP4
WHERE
    QTDAUDITORIA + QTDBUG + ESCOPO_ENTREGUE + QTDPRIORIDADE1 + QTDDEFEITOEMPRODUCAO
+ QTDDEFEITOEMHOMOLOGACAO > 0
```

Listagem 19 - Table input do transformation FATODESEMPENHOEQUIPERELEASE

Nas Figuras 17 e 18 estão um exemplo de como ficam os indicadores de análise de defeitos.

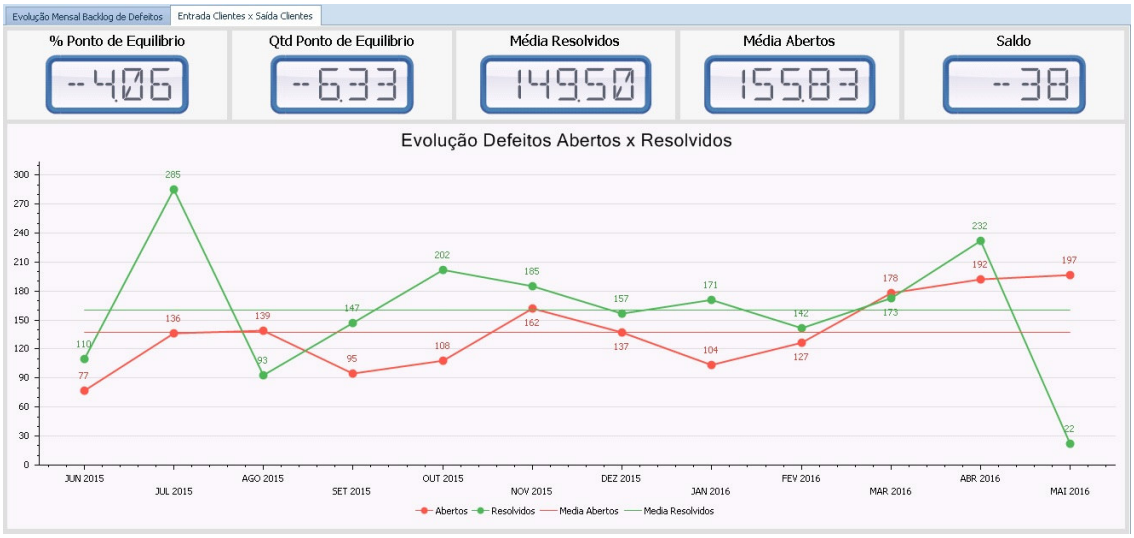

Figura 17 - Tela de análise de defeitos (a)

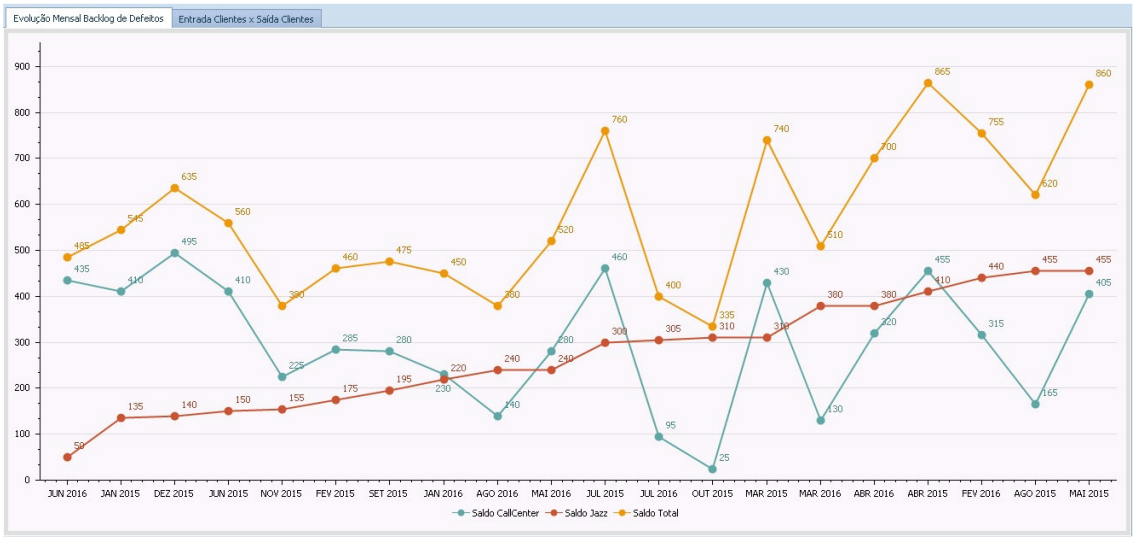

Figura 18 - Tela de análise de defeitos (b)

A instrução SQL do tipo select utilizada na ferramenta consulta SQL análise de defeitos - gráfico evolução mensal BACKLOG é apresentada na Listagem 20.

```
SELECT
    ORIGEMCLIENTECALLCENTER + ORIGEMCLIENTEJAZZ AS BACKLOGORIGEMCLIENTE,
    ORIGEMEMPRESACALLCENTER + ORIGEMEMPRESAJAZZ AS BACKLOGORIGEMEMPRESA,
    ORIGEMCLIENTECALLCENTER,
    ORIGEMEMPRESACALLCENTER,
    ORIGEMCLIENTECALLCENTER + ORIGEMEMPRESACALLCENTER AS BACKLOGCALLCENTER,
    ORIGEMCLIENTEJAZZ,
    ORIGEMEMPRESAJAZZ,
    ORIGEMCLIENTEJAZZ + ORIGEMEMPRESAJAZZ AS BACKLOGJAZZ,
    ORIGEMCLIENTECALLCENTER   +   ORIGEMEMPRESACALLCENTER   +   ORIGEMCLIENTEJAZZ   +
ORIGEMEMPRESAJAZZ AS BACKLOG,
    ABRMES ||' '|| NUMEROANO AS MES,
    FATO_SALDO_BACKLOG_MES.IDMES
FROM
    DBA.FATO_SALDO_BACKLOG_MES FATO_SALDO_BACKLOG_MES
    JOIN DBA.DIME_TEMPO_MES MES ON
         FATO_SALDO_BACKLOG_MES.IDMES = MES.IDMES
WHERE
    NUMEROANO >= YEAR(CURRENT DATE - 1 YEARS)
ORDER BY
    FATO_SALDO_BACKLOG_MES.IDMES
```

Listagem 20 - Instrução SQL do tipo select utilizada na ferramenta consulta SQL análise de defeitos

A Listagem 21 apresenta a instrução SQL do tipo select utilizada na ferramenta consulta SQL análise de defeitos - gráfico entrada clientes x saída fábrica.

```
SELECT
    QTDDEFEITOSABERTOS,
    QTDDEFEITOSRESOLVIDOS,
    CAST(
        CASE
            WHEN QTDMESESDEFEITOSABERTOS > 0 THEN
                QTDTOTALDEFEITOSABERTOS / QTDMESESDEFEITOSABERTOS
            ELSE
                0
        END AS DECIMAL(15,2)
    ) AS MEDIADEFEITOSABERTOS,
```

```
    CAST(
        CASE
            WHEN QTDMESESDEFEITOSRESOLVIDOS > 0 THEN
                QTDTOTALDEFEITOSRESOLVIDOS / QTDMESESDEFEITOSRESOLVIDOS
            ELSE
                0
            END AS DECIMAL(15,2)
    ) AS MEDIADEFEITOSRESOLVIDOS,
    MES
FROM
    (
    SELECT
        QTDDEFEITOSABERTOS,
        QTDDEFEITOSRESOLVIDOS,
        SUM(QTDDEFEITOSABERTOS) OVER() * 1.00 AS QTDTOTALDEFEITOSABERTOS,
        COUNT(1) OVER() * 1.00 AS QTDMESESDEFEITOSABERTOS,
        SUM(QTDDEFEITOSRESOLVIDOS) OVER() * 1.00 AS QTDTOTALDEFEITOSRESOLVIDOS,
        COUNT(1) OVER() * 1.00 AS QTDMESESDEFEITOSRESOLVIDOS,
        ABRMES || ' ' || NUMEROANO AS MES
    FROM
        DBA.FATO_CHAMADOS_ENTSAI_CLIENTE_MES FCEC
        JOIN DBA.DIME_TEMPO_MES MES ON
            FCEC.IDMES = MES.IDMES
    WHERE
        FCEC.IDMES >= DBA.YEARMONTHTOINT(SOMAMESES(CURRENT DATE, -8))
    ORDER BY
        NUMEROANO,
        NUMEROMES
    ) AS TMP
```

Listagem 21 - Select análise de defeitos: gráfico entrada clientes x saída fábrica

A instrução SQL do tipo select utilizada na ferramenta consulta SQL análise de defeitos - medidores entrada clientes x saída fábrica é apresentada na Listagem 22.

```
SELECT
    SUM(QTDDEFEITOSABERTOS) AS QTDTOTALDEFEITOSABERTOS,
    SUM(QTDDEFEITOSRESOLVIDOS) AS QTDTOTALDEFEITOSRESOLVIDOS,
    SUM(QTDDEFEITOSRESOLVIDOS) - SUM(QTDDEFEITOSABERTOS) AS SALDO,
    MEDIADEFEITOSABERTOS,
    MEDIADEFEITOSRESOLVIDOS,
    PERPONTOEQUILIBRIO,
    QTDPONTOEQUILIBRIO
FROM
    (
    SELECT
        QTDDEFEITOSABERTOS,
        QTDDEFEITOSRESOLVIDOS,
        MEDIADEFEITOSABERTOS,
        MEDIADEFEITOSRESOLVIDOS,
        CAST(
            CASE
                WHEN MEDIADEFEITOSABERTOS > 0 THEN
                    ( ( ( MEDIADEFEITOSRESOLVIDOS / MEDIADEFEITOSABERTOS ) * 100 )
- 100 )
                ELSE
                    0
            END AS DECIMAL(15,2)
        ) AS PERPONTOEQUILIBRIO,
        MEDIADEFEITOSRESOLVIDOS - MEDIADEFEITOSABERTOS AS QTDPONTOEQUILIBRIO
    FROM
        (
        SELECT
            QTDDEFEITOSABERTOS,
            QTDDEFEITOSRESOLVIDOS,
            CAST(
                CASE
```

```
                    WHEN QTDMESESDEFEITOSABERTOS > 0 THEN
                        QTDTOTALDEFEITOSABERTOS / QTDMESESDEFEITOSABERTOS
                    ELSE
                        0
                END AS DECIMAL(15,2)
            ) AS MEDIADEFEITOSABERTOS,
            CAST(
                CASE
                    WHEN QTDMESESDEFEITOSRESOLVIDOS > 0 THEN
                        QTDTOTALDEFEITOSRESOLVIDOS / QTDMESESDEFEITOSRESOLVIDOS
                    ELSE
                        0
                    END AS DECIMAL(15,2)
            ) AS MEDIADEFEITOSRESOLVIDOS
        FROM
            (
            SELECT
                QTDDEFEITOSABERTOS,
                QTDDEFEITOSRESOLVIDOS,
                SUM(QTDDEFEITOSABERTOS) OVER() * 1.00 AS QTDTOTALDEFEITOSABERTOS,
                COUNT(1) OVER() * 1.00 AS QTDMESESDEFEITOSABERTOS,
                SUM(QTDDEFEITOSRESOLVIDOS)       OVER()       *       1.00       AS
QTDTOTALDEFEITOSRESOLVIDOS,
                COUNT(1) OVER() * 1.00 AS QTDMESESDEFEITOSRESOLVIDOS
            FROM
                DBA.FATO_CHAMADOS_ENTSAI_CLIENTE_MES FCEC
            WHERE
                FCEC.IDMES >= DBA.YEARMONTHTOINT(SOMAMESES(CURRENT DATE, -8))
            ) AS TMP
        ) AS TMP2
    ) AS TMP3
GROUP BY
    MEDIADEFEITOSABERTOS,
    MEDIADEFEITOSRESOLVIDOS,
    PERPONTOEQUILIBRIO,
    QTDPONTOEQUILIBRIO
```

Listagem 22 - Select análise de defeitos: medidores entrada clientes x saída fábrica

As Figuras 19 e 20 apresentam um exemplo de como deve ficar o indicador de evolução de qualidade.

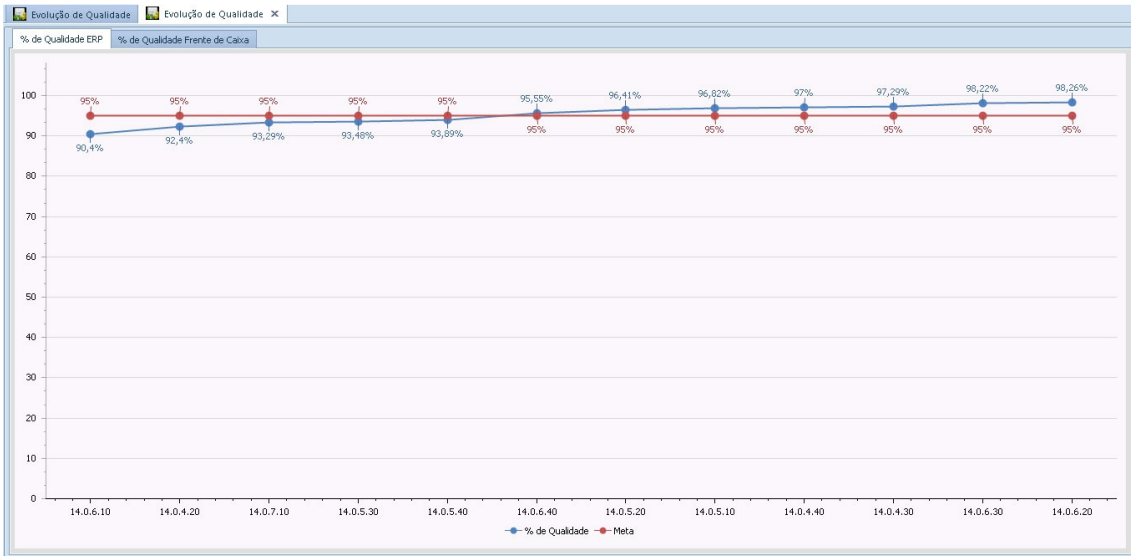

Figura 19 - Indicador de evolução de qualidade (a)

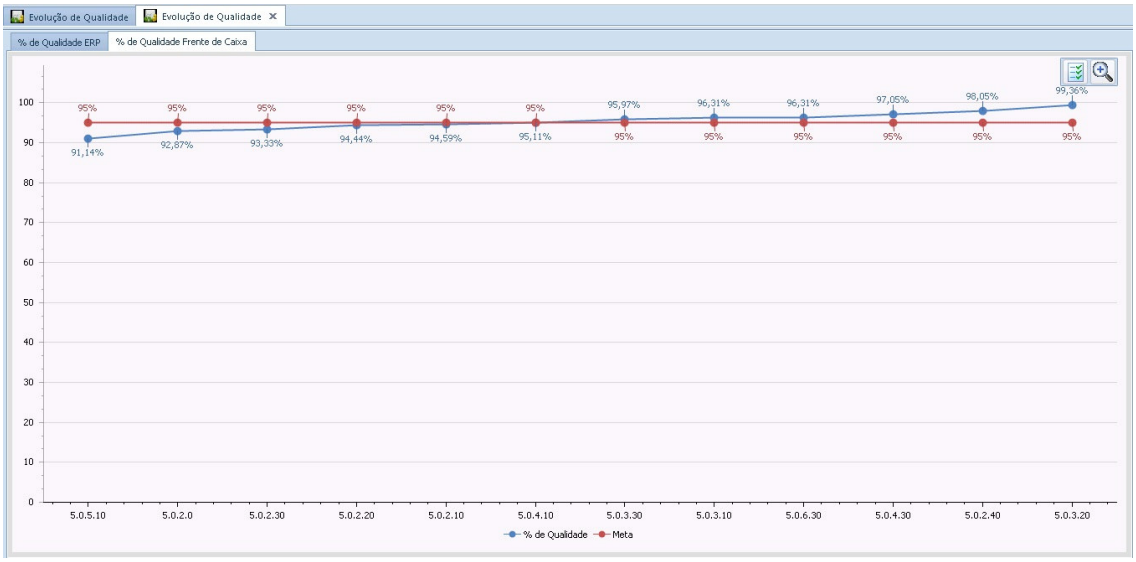

Figura 20 - Indicador de evolução de qualidade (b)

Na Listagem 23 está a instrução SQL do tipo select utilizada na ferramenta consulta SQL - evolução de qualidade - gráfico % de qualidade ERP.

```
SELECT
    IDVERSAO,
    DESCRVERSAO,
    PERQUALIDADE,
    95 AS PERMETAQUALIDADE
FROM
    (
    SELECT
        IDVERSAO,
        DESCRVERSAO,
        CASE
            WHEN QTDDEFEITOEMHOMOLOGACAO + QTDDEFEITOEMPRODUCAO + QTDBUG = 0 THEN
                0
            ELSE
                CAST(   100   *   (   1   -   (   QTDDEFEITOEMPRODUCAO   /   (   (
QTDDEFEITOEMHOMOLOGACAO  +  QTDDEFEITOEMPRODUCAO  +  QTDBUG  )  *  1.00  )  )  )  AS
DECIMAL(15,2))
        END AS PERQUALIDADE
    FROM
        (
        SELECT
            FATO_DESEMPENHO_EQUIPE_RELEASE.IDVERSAO,
            DESCRVERSAO,
            SUM(QTDBUG) AS QTDBUG,
            SUM(QTDDEFEITOEMPRODUCAO) AS QTDDEFEITOEMPRODUCAO,
            SUM(QTDDEFEITOEMHOMOLOGACAO) AS QTDDEFEITOEMHOMOLOGACAO
        FROM
            DBA.FATO_DESEMPENHO_EQUIPE_RELEASE FATO_DESEMPENHO_EQUIPE_RELEASE
            JOIN DBA.DIME_VERSOES DIME_VERSOES ON
                 DIME_VERSOES.IDVERSAO = FATO_DESEMPENHO_EQUIPE_RELEASE.IDVERSAO
        WHERE
            AREADEPROJETO = 'ERP'
        GROUP BY
            FATO_DESEMPENHO_EQUIPE_RELEASE.IDVERSAO,
            DESCRVERSAO
        ) AS TMP
    ORDER BY
        IDVERSAO DESC
    FETCH FIRST 12 ROWS ONLY
```

```
    ) AS TMP
ORDER BY
    IDVERSAO
```

Listagem 23 - Select evolução de qualidade: gráfico % de qualidade ERP

Na Listagem 24 estão uma instrução SQL do tipo select utilizada na ferramenta consulta SQL - evolução de qualidade - gráfico % de qualidade frente de caixa.

```
SELECT
    IDVERSAO,
    DESCRVERSAO,
    PERQUALIDADE,
    95 AS PERMETAQUALIDADE
FROM
    (
    SELECT
        IDVERSAO,
        DESCRVERSAO,
        CASE
            WHEN QTDDEFEITOEMHOMOLOGACAO + QTDDEFEITOEMPRODUCAO + QTDBUG = 0 THEN
                0
            ELSE
                CAST(   100   *   (   1   -   (   QTDDEFEITOEMPRODUCAO   /   (   (
QTDDEFEITOEMHOMOLOGACAO  +  QTDDEFEITOEMPRODUCAO  +  QTDBUG  )  *  1.00  )  )  )  AS
DECIMAL(15,2))
        END AS PERQUALIDADE
    FROM
        (
        SELECT
            FATO_DESEMPENHO_EQUIPE_RELEASE.IDVERSAO,
            DESCRVERSAO,
            SUM(QTDBUG) AS QTDBUG,
            SUM(QTDDEFEITOEMPRODUCAO) AS QTDDEFEITOEMPRODUCAO,
            SUM(QTDDEFEITOEMHOMOLOGACAO) AS QTDDEFEITOEMHOMOLOGACAO
        FROM
            DBA.FATO_DESEMPENHO_EQUIPE_RELEASE FATO_DESEMPENHO_EQUIPE_RELEASE
            JOIN DBA.DIME_VERSOES DIME_VERSOES ON
                 DIME_VERSOES.IDVERSAO = FATO_DESEMPENHO_EQUIPE_RELEASE.IDVERSAO
        WHERE
            AREADEPROJETO = 'FRENTEDECAIXA'
        GROUP BY
            FATO_DESEMPENHO_EQUIPE_RELEASE.IDVERSAO,
            DESCRVERSAO
        ) AS TMP
    ORDER BY
        IDVERSAO DESC
    FETCH FIRST 12 ROWS ONLY
    ) AS TMP
ORDER BY
    IDVERSAO
```

Listagem 24 - Gráfico % de qualidade frente de caixa

5 CONCLUSÃO

O processo de BI descrito neste trabalho obteve sucesso em extrair informações das aplicações IBM Smartcloud Control Desk, IBM Rational Team Concert e de um sistema desenvolvido pela própria Empresa para uso interno, um Call Center. Por meio desse processo foi possível criar indicadores que demonstrassem a qualidade e a produtividade da Empresa, servindo de apoio para tomada de decisão.

Apesar da dificuldade inicial de extrair informações do IBM Rational Team Concert, seja com conexão JDBC nativa ao banco de dados ou fazendo o consumo da API REST, com o uso do recurso de consulta com possibilidade de exportação para Excel em formato csv, a dificuldade inicialmente encontrada foi contornada. Isso porque, foi possível importar e tratar as informações utilizando a Kettle, antes de gravar os dados obtidos no Data Warehouse.

A Kettle, ferramenta open source e fácil de utilizar, possui diversos recursos, dos quais chamou a atenção a possibilidade de incluir códigos JavaScript e Java e a possibilidade de consumo de APIs. Entretanto, nota-se carência de informações disponíveis sobre como utilizar alguns dos recursos que essa ferramenta oferece.

O IBM DB2 LUW Express-C é simples de ser utilizado, não demandando preocupações com parametrização de memória, pois possui um recurso chamado *self tunning memory*. Esse recurso, quando ativado, realiza o ajuste automático de seus *pools* de memória. Mesmo sendo utilizada a versão comunidade da ferramenta IBM DB2 LUW Express-C, não foram encontradas restrições que pudessem gerar dificuldades no trabalho realizado. Contudo, se fosse possível ter acesso a algumas de suas *features*, melhorias poderiam ter sido realizadas.

A plataforma Vistra BI possibilitou o desenvolvimento de indicadores visando facilitar a interação com o usuário, fazendo o uso de seus gráficos altamente customizáveis. Essa ferramenta disponibiliza vários tipos de medidores disponíveis. Ela possui, ainda, um *e-learn* em seu portal. Esse é um aspecto muito positivo, pois acelera o aprendizado, mesmo sendo uma ferramenta do tipo *drag and drop* e intuitiva.

Como trabalhos futuros que possam ser desenvolvidos a partir deste apresentado, recomenda-se o estudo da relação entre custo e beneficio do *row compression* do IBM DB2. Esse recurso está disponível nas versões mais completas. E, ainda, a possibilidade de escrever código Java dentro da Kettle, que faça o consumo da API REST do IBM Rational Team Concert, sem a necessidade de realizar a importação de dados de planilhas Excel.

REFERÊNCIAS

CHAPMAN, Arthur D. **Principles data quality: global biodiversity information facility**. Universitetsparken Copenhagen Denmark, 2005.

CUZZOCREA, Alfredo; PUGLISI, Laura. **Record linkage in data warehousing: state-of-the-art analysis and research perspectives**. In: 22nd International Workshop on Database and Expert Systems Applications, 2011, p. 121-125.

JAZZ. Disponível em: <https://jazz.net/wiki/bin/view/Main/ReportsRESTAPI#Resources_provided_by_RTC>. Acesso em: 28 ago. 2016.

KANTARDZIC, Mehmed. **Data mining: concepts, models, methods, and algorithms**. 2 ed. Data - Mining Concepts. Institute of Electrical and Electronics Engineers, John Wiley & Sons, Inc., 2011.

MOHAJIR, Mohammed EI; LATRACHE, Amal. **Unifying and incorporating functional and non functional requirements in datawarehouse conceptual design**. IEEE, p. 49-57, 2012.

ORACLE. **Data warehousing concepts**. Disponível em: <https://docs.oracle.com/cd/B10500_01/server.920/a96520/concept.htm>. Acesso em: 09 fev. 2016.

SAVITRI, Fivien Nur; LAKSMIWATI, Hira. **Study of localized data cleansing process for ETL performance improvement in independent Datamart**. In: International Conference on Electrical Engineering and Informatics, Bandung, Indonesia, 2011, p. 1-6.