DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Implantação de Ferramenta Open Source de Business Intelligence para Gerenciamento de Clientes com Grande Número de Usuários

Relatório submetido à Universidade Federal de Santa Catarina

como requisito para a aprovação na disciplina

DAS 5511: Projeto de Fim de Curso

Daniel Valenti

Florianópolis, Agosto de 2016

Implantação de Ferramenta Open Source de Business Intelligence para Gerenciamento de Clientes com Grande Número de Usuários

Daniel Valenti

Esta monografia foi julgada no contexto da disciplina
DAS5511: Projeto de Fim de Curso
e aprovada na sua forma final pelo
Curso de Engenharia de Controle e Automação

Prof. *Ricardo José Rabelo*

_____
Assinatura do Orientador

Banca Examinadora:

Felipe Santos Eberhardt
Orientador na Empresa

Prof. Ricardo José Rabelo
Orientador no Curso

Luciano Barreto
Avaliador

Fabio Henriques Mantelli
Vinicius de Morais Justo
Debatedores

Resumo

Com o grande avanço da tecnologia, tornou-se possível a aquisição de dados sobre negócios de forma mais fácil. Com isso, grandes quantidades de informações são armazenadas diariamente, o que ajuda as pessoas de negócio a terem um melhor conhecimento do desempenho de suas empresas. Entretanto, a abundância de geração de dados faz com que as análises necessárias demandem tempo demasiado para obtenção de resultados relevantes, podendo assim, ocasionar em uma tomada de decisão tardia. Com o objetivo de solucionar esse problema, foram desenvolvidos sistemas de *Business Intelligence*, que têm por finalidade auxiliar usuários de negócios em suas tomadas de decisão. Sistemas de BI são desenvolvidos para analisar uma grande quantidade de dados e apresentar de forma simples e amigável as informações mais relevantes para a gerência. Essa exposição de informação é feita através de painéis, por exemplo, contendo gráficos e tabelas que o usuário pode interagir de forma fácil e obter as informações precisas de forma rápida. O presente documento descreve o projeto desenvolvido para implantação de um sistema de BI para a empresa Portobello Shop para auxiliar os gerentes e franqueados na administração de suas lojas.

Abstract

With the technology evolution, it became possible to acquire business data in an easier way. Thereby, large amounts of information are stored daily, which helps business people to have a better understanding of the performance of their companies. However, the abundance of data generation makes the time to analyze data too long to obtain relevant results, and may thus result in a delayed decision. In order to solve this problem, business intelligence systems have been developed. These systems are designed to help business users in their decision making. BI systems are designed to analyze a lot of data and present it in a simple and friendly way, showing the most relevant information for management. This information is displayed through panels, for example, containing charts and tables that the users can interact easily and get the accurate information quickly. This document describes the project developed to implement a BI system for the Portobello Shop Company to assist managers and franchisees in the management of theirs stores.

Sumário

Simbologia

A seguir:

BI – Business Intelligence

CSS – Cascading Style Sheets

DW – Data Warehouse

FK – Foreign Key

JS – JavaScript

OLAP – On-Line Analytical Processing

PDI – Pentaho Data Integration

PK – Primary Key

SQL – Structured Query Language

Capítulo 1: Introdução

1.1: Introdução à problemática

Atualmente, o ambiente de negócios no qual as organizações operam tem ficado mais complexo e variante. Desse modo, as empresas privadas, principalmente, vêm se sentindo cada vez mais pressionadas a responderem questões de inovação em seus modos de operação.

A frequente necessidade de inovação obriga essas empresas a tomarem decisões de maneira rápida e frequente, tanto para questões simples quanto para questões muito complexas. Para isso, algumas decisões exigem análises de grande quantidade de dados, além de informações e conhecimento.

A geração de grande quantidade de relatórios e dados sem direção resulta em análises que demandam um dispendioso tempo de trabalho para obter um resultado que possa ser utilizado nas tomadas de decisão. Com a demora desses resultados, a gerência, muitas vezes, não é capaz de definir as ações a serem realizadas em tempo hábil.

Além do problema do tempo necessário para obter esses resultados, muitas vezes, sistemas de relatórios sobrepostos podem fornecer dados duvidosos, posto que, os dados utilizados podem ter origens diferentes. Consequentemente, não é possível tomar decisões em tempo suficiente, visto que não se sabem quais dados são mais precisos.

1.2: Problemática

Atualmente, a empresa Portobello Shop é responsável por mais de 140 lojas, incluindo franquias e lojas próprias. Essas lojas possuem um sistema responsável por armazenar diversas informações sobre as vendas realizadas, gerando uma grande quantidade dados.

Para que esses estabelecimentos funcionem de forma otimizada, a análise das informações geradas é uma etapa essencial. Porém, a grande quantidade de dados armazenados dificulta a visualização de eventos de maior importância para o gerenciamento da organização.

Atualmente, as análises desses dados gerados diariamente são feitas através de cruzamento de tabelas. Com isso, é possível obter os indicadores desejados para que se tenha um bom controle da empresa.

Porém, devido a geração de inúmeras tabelas e dados, os responsáveis por tomarem decisões importantes necessitam de muito tempo para analisar os dados e decidirem os passos a serem seguidos.

Como consequência, eventualmente, o dispendioso tempo gasto para realizar essas análises ocasiona a perda do *timing* para a solução de problemas relevantes. Com isso, abre-se espaço para a concorrência crescer e consequentemente trazer grandes prejuízos para a organização.

1.3: Objetivos

Este projeto tem como objetivo avaliar a viabilidade e escalabilidade da implantação de uma ferramenta de *Business Intelligence* para clientes com um grande número de usuários. Por exemplo, uma empresa (cliente) com muitos funcionários (usuários), onde cada funcionário terá seu próprio login e senha para acessar o sistema.

1.3.1: Geral

O objetivo do presente trabalho foi realizar a implantação de um sistema de BI (Business Intelligence) para a gerência das lojas da Portobello Shop. A ferramenta tem por objetivo propiciar um painel de controle para a gestão integrada das lojas da rede. Essa ferramenta conta com diversos indicadores, que medem o desempenho de cada loja e seus funcionários, com isso, possibilitando a tomada de decisões importantes para a melhoria do desempenho da loja.

A utilização de uma ferramenta de BI reduz drasticamente o tempo investido nas análises de dados, em razão de um dos seus objetivos ser a exibição de dados mais importantes e de forma interativa para o usuário. Consequentemente, o tempo utilizado atualmente para realizar essas análises pode ser utilizado efetivamente para as tomadas de decisão. Por fim, a aplicação desse sistema possibilita tornar a empresa mais preparada para lidar com a concorrência.

A gestão por meio de indicadores de desempenho é o ponto de partida para uma empresa obter uma melhoria em seu desempenho, pois, o que não é medido, não é gerenciado; o que não é gerenciado, não pode ser melhorado; e se não pode manter um constante processo de melhorias, em breve deixará de existir. Sendo assim, implementa-se um sistema de BI para auxiliar o gerenciamento das lojas Portobello Shop. [1]

1.3.2: Específicos

- Selecionar indicadores e métricas de desempenho a serem utilizados na ferramenta de BI;
- Modelar o Banco de Dados;
- Desenvolver o processo de ETL (Extração, Transformação e Carga);
- Criar um Cubo de Dados;
- Criar um *Dashboard* para a visualização dos dados;
- Adaptar o *Dashboard* para versão mobile;
- Liberar o BI para as lojas;

1.4: Estrutura do Documento

Este documento é dividido em 7 capítulos. O capítulo 2 apresenta as instituições onde foi realizado o presente projeto e seus processos. A solução do problema é apresentado no capítulo seguinte, juntamente com as ferramentas disponíveis para implantação da solução. No capítulo 4 são apresentados os fundamentos teóricos que serão utilizados no decorrer do projeto. A implantação do

sistema desenvolvido é exposto no capítulo 5, detalhando todos os procedimentos realizados neste projeto. O capítulo 6 expõe os resultados obtidos na realização deste projeto. Por último, são apresentadas conclusões e possíveis melhorias a serem feitas no projeto desenvolvido.

Capítulo 2: Institucional

O presente projeto foi desenvolvido na empresa FernBI focado na implantação de um sistema de BI utilizando uma ferramenta diferente do Qlikview, o qual é a ferramenta utilizada atualmente pela empresa. O sistema a ser implantado tem como objetivo auxiliar um dos principais clientes da FernBI, a Portobello, nas suas tomadas de decisão.

2.1: FernBI

Situada em Florianópolis, a FernBI é uma empresa especializada em soluções de inteligência de negócios e tecnologia. A empresa tem como foco principal o desenvolvimento de ferramentas de BI, que auxilia os clientes a terem *insights* de melhorias na gestão de seus processos, na detecção de *gaps*, no acompanhamento de indicadores e nas tomadas de decisão assertivas em tempo reduzido.

Focada não somente em tecnologia, a empresa possui uma equipe altamente capacitada também na área de consultoria de negócios e otimização/automatização de processos de gestão.

A FernBI é responsável pelo desenvolvimento de grande parte do sistema de BI utilizado atualmente na empresa Portobello. Desse modo, a Portobello é um dos principais clientes da FernBI.

2.2: Portobello

A empresa Portobello S.A., fundada em 1979, situa-se no município de Tijucas, em Santa Catarina. Sua área construída é de 205 mil metros quadrados, sendo assim, considerada a maior empresa cerâmica do Brasil atualmente. Sua receita bruta é superior a R$ 1 bilhão, o que torna a empresa bastante importante para a região. A produção anual supera os 30 milhões de metros quadrados,

atendendo mais de 65 países, abrangendo os cinco continentes e também o mercado interno.

O presente projeto foi realizado com o objetivo de atender o cliente Portobello Shop. Esse canal comercial é uma franqueadora responsável pela venda e distribuição dos produtos para uma rede de mais de 140 lojas.

Atualmente, a Portobello Shop conta com um sistema de BI utilizado para o gerenciamento geral da rede de franquias. Contudo, devido à utilização de um software de BI que necessita licenciamento, o acesso ao sistema fica restrito somente a alguns funcionários da empresa Portobello Shop.

Em razão disso, pessoas responsáveis pelo gerenciamento de uma franquia não possuem acesso às informações relevantes de maneira rápida e eficiente, visto que não contam com o acesso ao sistema de BI. Consequentemente, gerentes e franqueados necessitam utilizar outros meios para analisar os dados gerados por suas lojas. Geralmente utilizando tabelas de Excel e realizando cruzamento de tabelas, consumindo, assim, muito tempo e atrasando as tomadas de decisão de gerenciamento.

Capítulo 3: A Solução

Com o intuito de auxiliar os gerentes e franqueados das lojas Portobello Shop, a implantação de um sistema de *Business Intelligence* foi definida como solução, uma vez que esse sistema é responsável por organizar os dados gerados pelas lojas de forma mais clara e amigável. Para isso, são utilizados gráficos e tabelas contendo as informações mais importantes para o gerenciamento.

3.1: *Business Intelligence*

A competição na área de negócios vem crescendo continuamente. Sendo assim, a necessidade de um acesso rápido e eficiente às informações de negócios torna-se indispensável. Para isso, são utilizados sistemas de Inteligência de Negócios (ou *Business Intelligence)*, os quais utilizam tecnologia e produtos para fornecerem informações necessárias para as tomadas de decisão estratégica.

Traduzido e adaptado do livro "*Getting Started with DataWarehouse and Business Intelligence*", Inteligência de Negócios significa "utilizar seus dados ativos para tomar melhores decisões de negócios. Trata-se de acesso, análise, e revelação de novas oportunidades."

Os sistemas de BI utilizam *Data Warehouses* (DW) como fonte de dados. Esses são responsáveis por armazenar todos os dados gerados em apenas um local, facilitando assim, a organização e os trabalhos que serão realizados futuramente pelo sistema de BI. Os DWs armazenam também os históricos dos dados, possibilitando o trabalhar com diferentes versões de informações.

Com os dados armazenados no DW, a maioria dos sistemas de BI utiliza ferramentas de cubo de dados. Os cubos de dados OLAP (Online Analytical Processing) são implementados em banco dados multidimensionais. Esses cubos armazenam e indexam os dados usando formatos de dados dimensionais. Dessa forma, a pesquisa de dados em um cubo OLAP ocorre de forma mais rápida e

eficiente devido aos pré-cálculos, estratégias de indexação, e outras otimizações que o cubo utiliza. [3]

O sistema de BI acessa, então, o cubo de dados e realiza as pesquisas necessárias para que sejam adquiridas as informações desejadas para a gerência da empresa. Essas informações são exibidas através de painéis, conhecidos como Dashboards, de forma clara e interativa. Esses dashboards são compostos geralmente por tabelas, gráficos e indicativos dos dados mais relevantes da organização.

O acesso do usuário final se dá através da utilização de uma página web, na qual o cliente pode interagir com o Dashboard e selecionar as informações desejadas. Além disso, é possível restringir o tipo de acesso que cada usuário terá, tornando assim, o sistema de BI um método seguro de trabalho. Pois, desse modo, é possível definir quais usuários poderão ver quais informações.

3.1.1: A Evolução dos Sistemas de Informação de Negócios

O livro "*Getting Started with DataWarehouse and Business Intelligence*" da IBM, divide os sistemas de informação de negócios em três gerações: *Host-Based Query and Reporting*, *Data Warehousing* e *Business Intelligence*.

3.1.1.1: Primeira Geração: *Host-Based Query and Reporting*

No início dos sistemas de informação de negócios, utilizavam-se aplicações para fornecer, aos gerentes, dados que eram necessários. Geralmente, os resultados dessas aplicações continham muitas informações, totalizando um grande volume de folhas. Desse modo, os gerentes tinham que pesquisar as informações que realmente queriam.

Estes sistemas necessitavam de um bom conhecimento em computadores e análise de dados. Por isso, geralmente, eram apenas analistas de dados que utilizavam esses programas e então enviavam os dados para os usuários de negócios. [1]

3.1.1.2: Segunda Geração: *Data Warehousing*

A segunda geração conta com os sistemas de *Data Warehousing* que possuem algumas vantagens em relação à primeira geração. Esses sistemas são organizados de uma forma mais clara, facilitando o entendimento dos usuários de negócios. Além de tornar possível o armazenamento de dados históricos.

Alguns sistemas de *Data Warehousing* disponibilizam ferramentas que auxiliam na tomada de decisão, entretanto tinham como foco principal, o desenvolvimento de tecnologia. [1]

3.1.1.3: Terceira Geração: *Business Intelligence*

Ao contrário dos sistemas de *Data Warehousing*, sistemas de *Business Intelligence* concentram suas atenções em ferramentas que auxiliem os usuários de negócios em suas tomadas de decisão. A segunda geração ainda tinha como objetivo principal o armazenamento dos dados, já essa, tem como foco facilitar o acesso aos dados.

Para isso, investiu-se em desenvolvimento de ferramentas gráficas avançadas e também sistemas de processamento analítico online (OLAP), que será explicado na seção 4.3: deste documento.

Um dos principais objetivos dos sistemas de *BI* é facilitar o acesso aos dados, possibilitando assim, os usuários responderem as principais perguntas de negócios geradas diariamente. [1]

3.2: Objetivos dos Sistemas de BI

Para que os sistemas de BI sejam eficientes, alguns requisitos devem ser atendidos. Dessa forma:

- **O sistema de BI deve tornar as informações facilmente acessíveis.** Os conteúdos do sistema de BI devem ser de fácil entendimento. Os dados devem ser intuitivos e óbvios para os usuários de negócios. As

ferramentas de BI devem ser simples e fáceis de utilizar. Também, devem trazer as informações solicitadas pelos usuários no menor tempo possível. Ou seja, um sistema de BI deve ser simples e rápido.

- **O sistema de BI deve apresentar informações de forma consistente.** Os dados do sistema de BI devem ser confiáveis, devem ser cuidadosamente agregados de diferentes fontes, limpos, ter qualidade assegurada, e somente serem entregues aos usuários quando estiverem prontos para serem consumidos.
- **O sistema de BI deve ser adaptável a mudanças.** Necessidades do usuário, condições de negócios, dados e tecnologia são sujeitos a mudanças. O sistema de BI deve ser desenvolvido para que se adapte a essas mudanças sem invalidar dados existentes ou aplicações.
- **O sistema de BI deve apresentar as informações a tempo.** Como os sistemas de BI são utilizados de forma mais intensiva para decisões operacionais, os dados gerados devem ser convertidos e disponíveis para o usuário em um curto período de tempo.
- **O sistema de BI deve proteger as informações disponíveis.** Algumas informações valiosas da organização são guardadas nos *Data Warehouses*. Sendo assim, essas informações devem estar protegidas para que não sejam acessadas por pessoas não autorizadas.
- **O sistema de BI deve servir como uma base competente e confiável para tomadas de decisão.** O sistema de BI deve possuir os dados certos para auxiliar na tomada de decisões. A funcionalidade mais importante de um sistema de BI é apresentar importantes análises de informações necessárias e confiáveis para tomadas de decisão. [3]

3.3: As Ferramentas Disponíveis

Após definida a implantação de um sistema de BI, uma pesquisa foi realizada com a finalidade de selecionar a ferramenta mais adequada para atender as especificações do cliente.

Atualmente, a empresa FernBI desenvolve sistemas de BI utilizando ferramentas da Qlik, principalmente o QlikView. Sendo assim, os funcionários da empresa já são familiarizados com a ferramenta e possuem um ótimo conhecimento em desenvolvimento de sistemas de BI com ferramentas Qlik. [9]

Porém, para utilizar as ferramentas oferecidas pela Qlik, é necessário comprar licenças para cada usuário. Assim sendo, como o objetivo do projeto é desenvolver um sistema de BI para que vários usuários tenham acesso as informações, foram estudadas as ferramentas presentes na suíte Pentaho [10], visto que essas ferramentas são de código aberto e não necessitam de licença.

3.3.1: QlikView

QlikView é um software de BI desenvolvido pela empresa Qlik. Essa ferramenta é utilizada atualmente pela empresa Portobello Shop. Entretanto, não são todas as pessoas que podem acessá-la, visto que é necessário adquirir uma licença para cada usuário, ocasionando o aumento do custo para empresa.

A alocação dos dados em memória RAM é a principal vantagem dessa ferramenta. Por essa razão, o Qlikview não utiliza a tecnologia OLAP e sim, uma tecnologia chamada AQL. Essa tecnologia utiliza uma tabela fato a qual todas as outras tabelas são associadas. O sistema possibilita a diminuição do tempo de projeto e, também, o aumento da velocidade de resposta do sistema. [8]

3.3.2: Pentaho

Pentaho é um software de código aberto de inteligência de negócios desenvolvido em Java pela Pentaho Corporation. Com esse programa, é possível desenvolver um sistema BI, incluindo os processos de extração dos dados, criação de cubo de dados, criação de Dashboards entre outras funcionalidades. Dentre os

componentes que o Pentaho possui, os seguintes elementos são suficientes para a solução do problema proposto e foram estudados:

- Pentaho Data Integration (PDI): ferramenta de extração, transformação e carga dos dados;
- Pentaho Schema Workbench: ferramenta de criação de cubo OLAP;
- Pentaho BI Server: ferramenta de interação com o usuário.

Para o desenvolvimento de um sistema de BI em Pentaho é necessário um conhecimento básico de programação nas linguagens JavaScript, HTML, CSS e MDX.

3.3.3: A escolha da Ferramenta

Se comparadas as duas ferramentas, QlikView e Pentaho, o desenvolvimento de um sistema de BI é muito menos trabalhoso utilizando QlikView, podendo chegar a gastar ¼ do tempo de desenvolvimento de um sistema equivalente em Pentaho. Todavia, como o cliente necessita de um grande número de usuários, foi definido como ferramenta do projeto o software Pentaho, pois não será necessário investir em licença, reduzindo assim, o custo de implantação do sistema de BI.

Capítulo 4: Fundamentos Teóricos e Softwares Utilizados

Este capítulo tem como objetivo fornecer o embasamento teórico necessário para um melhor entendimento do que foi realizado nesse projeto. Serão explicados nesse capítulo conceitos utilizados sobre Modelagem do Banco de Dados, ETL e Cubo de Dados.

Como abordado no capítulo anterior, esse projeto foi desenvolvido utilizando as ferramentas da Pentaho Corporation, entre elas estão: Pentaho Data Integration, Schema Workbench e Pentaho BI Server, utilizadas para desenvolver os processos de ETL, criação do cubo de dados e o servidor BI, respectivamente. O funcionamento e aplicação dessas ferramentas fazem parte do escopo deste capítulo.

4.1: Modelagem do Banco de Dados

A modelagem dimensional tem como objetivo organizar o banco de dados de forma simples. Essa simplicidade deve atender tanto a facilidade com que os usuários entenderão as informações, como também, deve organizar o banco de dados de modo que os softwares consigam buscar as respostas de forma rápida e eficiente. [3]

Nesse projeto, será utilizado o conceito Esquema Estrela (ou *Star Schema* em inglês). Essa metodologia propõe uma visão para a modelagem de banco de dados para sistemas de apoio de decisão. Essa estrutura possui como principal característica a presença de dados altamente redundantes, melhorando, assim, o desempenho do sistema. [3]

O modelo Esquema Estrela possui dois componentes principais: a Tabela Fato e as Tabelas Dimensões.

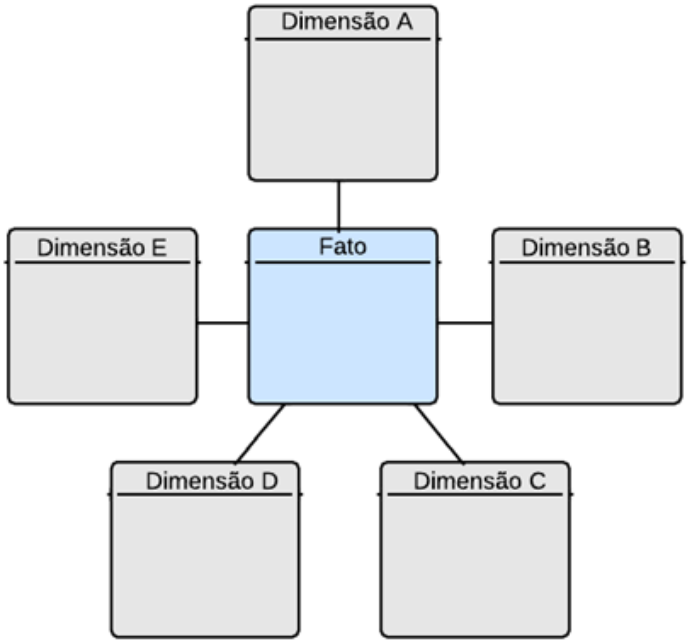

Figura 1 - Esquema Estrela

4.1.1: Tabela Fato

O termo “Fato” representa uma medida de negócios. A tabela fato tem como função armazenar as medidas de desempenho resultantes dos eventos de processos de negócios de uma organização. [3]

Na tabela Fato, cada linha corresponde a um evento de medição. Os dados de cada linha estão em um nível de detalhamento específico, chamado de grão (em inglês *grain*), como um produto vendido em uma determinada transação. Um dos princípios mais importantes, para a criação de uma tabela fato, é deixar todas as linhas no mesmo grão. Dessa forma, mantendo um único nível de detalhamento, garante-se que uma medida não se repita ao longo da tabela. [3]

Geralmente, os dados contidos na tabela fato são numéricos e aditivos, bem como o valor de um produto em reais. A aditividade é uma propriedade muito importante, visto que, usualmente, sistemas de BI trazem centenas, milhares e até mesmo milhões de linhas. Desse modo, é de grande utilidade poderem-se somar os dados. [3]

Em teoria, é possível que um fato seja um texto, no entanto, não é comum, posto que, não é possível a realização de cálculos (soma, média, etc.) com valores de texto, além ocuparem grande espaço de armazenamento.

Todas as tabelas fato possuem no mínimo duas ou mais *Foreign Keys* ou FK (Chaves Estrangeiras), que se referem às *Primary Keys* ou PK (Chaves Primárias) das tabelas Dimensão. Desse modo, a conexão entre a tabela fato e suas dimensões é estabelecida por meio dessas chaves. [3]

4.1.2: Tabelas Dimensão

As tabelas dimensão compreendem os valores de texto que estão associados aos eventos contidos na tabela Fato, eles descrevem “quem, o que, onde, quando, como e por que” associados a cada evento. É comum que essas tabelas, diferentemente das tabelas fato, possuam grande número de colunas ou atributos. Cada dimensão é definida por uma PK única, que funciona com referência para a tabela fato. [3]

A redundância de certas informações pode estar presente em uma tabela dimensão, entretanto, não há problema e não se devem normalizar essas informações em outra tabela. Essa normalização é chamada *Snowflaking* (em português: Floco de Neve) devido à modelagem das tabelas semelhar-se ao formato de um floco de neve. Por serem menores que a tabela fato, não é vantagem normalizar as informações contidas em uma tabela Dimensão, convém manter a simplicidade e acessibilidade. [3]

4.2: ETL

O processo de Extração, Transformação e Carga (ou em inglês *Extract, Transform and Load*) é a parte do sistema de BI que encontra-se entre os sistemas operacionais de origem e a área de apresentação do BI.

A extração é a primeira etapa do processo, na qual ocorre a obtenção dos dados para o *Data Warehouse* (DW). Isto é, extrair significa ler os dados existentes e armazenar as informações desejadas para que possam ser manipuladas nas seguintes etapas. [3]

Após a extração das informações desejadas, inicia-se o processo de transformação dos dados adquiridos. Essa etapa é responsável por realizar alterações necessárias nas informações para que essas estejam de acordo com a necessidade do sistema de BI. [3]

A etapa final, de carga, é responsável por carregar as informações com suas devidas alterações para o DW. Isto é, esse passo insere os dados em suas determinadas tabelas dimensão e fato para que sejam posteriormente utilizados no sistema de BI. [3]

Para desenvolver os processos de ETL desse projeto, foi utilizado o software Pentaho Data Integration (PDI), também conhecido como Kettle. O desenvolvimento de um processo de ETL nesse software se dá mediante a utilização de *steps*, os quais são responsáveis por realizarem as alterações nos dados adquiridos, como mostra a Figura 2.

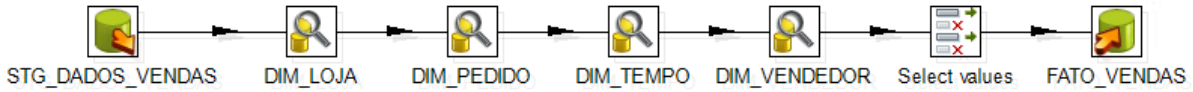

Figura 2 - ETL População Tabela Fato Vendas

Para realizar o processo de ETL utilizando o PDI, são criadas transformações (ou *transformations* em inglês). As transformações são compostas de *steps* responsáveis por realizar a extração, transformação e a carga dos dados desejados.

Depois de desenvolvidas as transformações, cria-se um *job*, como mostra a Figura 3. Este é responsável por organizar a ordem em que as transformações serão executadas, assim como definir o período que será executado. Dessa forma, torna-se um processo de ETL automático.

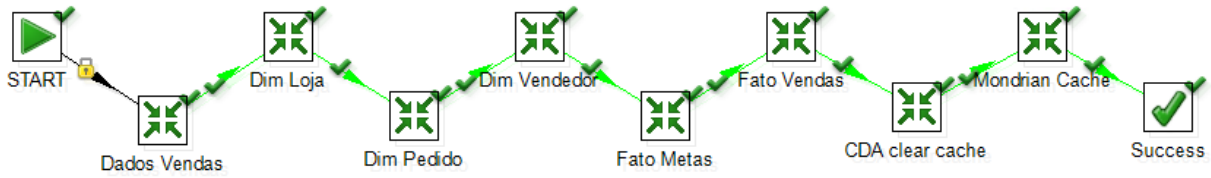

Figura 3 - Job ETL

4.3: Cubo de Dados

O cubo de dados OLAP (*On-Line Analytical Processing*) é um tipo de tecnologia de software que auxilia o usuário na compreensão de dados da organização, disponibilizando os dados de forma rápida, consistente e interativa. [4]

O conceito de OLAP inclui a noção ou ideia de dimensões com múltiplas hierarquias. Pode ser utilizado em diversos âmbitos, como na área de economia, abrangendo desde microagentes até macroeconomia ou, na área de material, abrangendo desde partículas atômicas até uma escala galáctica. [5]

Um cubo OLAP é constituído por Medidas (*Measures*) e Dimensões (*Dimensions*), que são ligadas através de uma tabela fato, assim como explicado na seção 4.1: sobre modelagem dimensional. [6]

Cada medida possui um nome, uma coluna na tabela Fato e um agregador. Em geral, utiliza-se “soma” como agregador, porém, “contador”, “mínimo”, “máximo” e “média” também são utilizados. [6]

Um membro é um ponto determinado em uma dimensão por um conjunto particular de valores. Hierarquia é um conjunto de membros organizados em uma estrutura conveniente para análise. Já, um nível é uma coleção de membros que possuem a mesma distância da raiz da hierarquia. Além disso, dimensão é definida como uma coleção de hierarquias que são definidas em uma mesma coluna da tabela Fato. [6]

Nesse projeto foi utilizado o software *Schema Workbench* desenvolvido pela Pentaho com a finalidade de criar o cubo de dados. A utilização desse software se dá através conexão com o banco de dados e o servidor BI.

Dessa forma, cria-se um *Schema* no qual estarão contidos os cubos de dados. Ao criar um cubo de dados, define-se qual será a tabela Fato a qual esse estará conectado. Em seguida, criam-se as dimensões que serão utilizadas e então, definem-se as tabelas do banco de dados na qual as informações de cada dimensão estão contidas.

Para cada dimensão criada, é possível definir seus níveis, com o propósito de, no momento da realização da consulta ao banco de dados, somente as

informações desejadas sejam trazidas para o servidor BI. Desse modo, diminui-se o tempo de pesquisa e melhora os resultados a serem exibidos.

Além da criação do cubo, o software *Schema Workbench*, é responsável por criar os *Roles* (ou cargos). Um cargo é responsável por definir quais as informações que poderão ser acessadas por cada usuário.

Ao criar um cargo, é necessário definir diversos componentes para que sejam limitadas as informações que cada usuário terá acesso. Os seguintes componentes devem ser definidos:

Role Name: define o nome do cargo que está sendo criado;

SchemaGrant: define o acesso padrão para as informações contidas no schema. Este pode ser definido como “all”, o que libera acesso a todas as informações do schema, ou “none”, que bloqueia o acesso a todas as informações. Esse acesso pode ser sobrescrito por objetos específicos.

CubeGrant: define o acesso a um cubo específico. Assim como o “SchemaGrant”, pode ser definido como “all” ou “none”, e pode ser sobrescrito por sub-objetos no cubo.

HyerarchyGrant: define o acesso a uma hierarquia. Esse acesso pode ser “all”, que significa que todos os membros da hierarquia podem ser acessados, “none”, que significa que a hierarquia será omitida do usuário, ou “custom”. O acesso “custom” permite definir um “topLevel”, que determina qual é o nível mais alto que pode ser visto pelo usuário (prevenindo o usuário de obter uma visão macro), um “bottomLevel”, que define o nível mais baixo que o usuário terá acesso (impedindo-o de ter acesso a detalhes mais específicos), ou controlar os membros que o usuário terá acesso definindo elementos em **MemberGrant.**

MemberGrant: libera (ou remove) o acesso a um determinado membro e todos os seus filhos. **MemberGrant** só pode ser definido se **HyerarchyGrant** estiver definido como “custom”.

As seguintes regras se aplicam na criação de *roles*:

1. **Members herdam acesso de seus pais.** Se for negado acesso à Santa Catarina, não será possível ver Florianópolis;

2. **Grants dependem da ordem.** Se for garantido acesso ao Brasil, e negado acesso à Santa Catarina, não será possível ver Santa Catarina nem Florianópolis. Porém, se for negado acesso à Santa Catarina e depois liberar acesso ao Brasil, será possível ver efetivamente tudo;
3. **Um Member é visível se algum de seus filhos for visível.** Se for negado acesso ao Brasil e liberado acesso à Santa Catarina, será possível ver Brasil e Santa Catarina, mas nenhum dos outros estados. Contudo, os totais referentes ao Brasil continuaram refletindo todos os estados.

MemberGrants não sobrescrevem "topLevel" e "bottomLevel" da hierarquia. Se for definido *topLevel* como "Estado" e liberar acesso à Santa Catarina, não será possível ver as informações referentes ao Brasil.

4.4: Pentaho BI Server

O Pentaho BI Server é a ferramenta que relaciona o usuário cliente às informações do cubo de dados e do banco de dados. Essa é uma ferramenta web, ou seja, acessada por um navegador de internet. A partir do login realizado nessa ferramenta, é possível ter acesso aos dados disponíveis. Dependendo das restrições de cada usuário, é possível somente a visualização das informações ou também a edição de painéis.

Os dashboards são painéis criados no servidor BI para que os usuários possam acessar as informações de forma mais fácil e rápida. Esses dashboards podem incluir diversos componentes, como gráficos de linha, gráfico de barra, gráficos de pizza, tabela entre outros.

O acesso ao servidor via web permite com que múltiplos usuários acessem as informações simultaneamente sem que um usuário interfira na exibição do outro. Dessa forma, no escopo desse projeto, é utilizado apenas um servidor e esse tem como objetivo atender todas as lojas da rede Portobello Shop simultaneamente.

Capítulo 5: Implantação do Sistema Desenvolvido

Este capítulo tem como objetivo descrever o modo no qual o sistema de BI foi desenvolvido. Será abordado, na prática, os conceitos e teorias explicados no Capítulo 4:. Trata-se, então, da modelagem do Dashboard; da modelagem do banco de dados; do desenvolvimento do processo de extração, transformação e carga; da conexão do servidor de BI com o banco de dados; da criação do cubo de dados; do desenvolvimento do Dashboard; da criação do processo de atualização diária automática; e da criação e restrição de acessos dos usuários aos dados.

5.1: Modelagem do Dashboard

Para o início do projeto, foi necessário modelar o painel principal desejado pelo cliente. Para isso, foi feita uma reunião para definir qual seria o produto final a ser entregue.

Sendo o foco principal do projeto, auxiliar o controle das vendas das lojas da Portobello, foram escolhidas as informações essenciais que deveriam estar contidas no Dashboard, sendo elas:

VNP – Venda Na Ponta é uma medida utilizada que informa o valor final faturado no pedido;

Desconto – É uma medida que informa o valor, em reais, do desconto dado em determinado pedido;

Ticket Médio – Ticket médio informa o valor médio dos pedidos realizados tanto para um determinado vendedor, como para uma loja;

Meta – É o valor de VNP o qual o vendedor ou loja deve atingir ao final de cada mês;

Número de Pedidos – Informa quantos pedidos foram realizados no período selecionado;

Vendedor – É o nome do vendedor que realizou o pedido;

Loja – Nome da loja onde foi realizado o pedido;

Ano – Ano em que o pedido foi realizado;

Mês – Mês em que o pedido foi realizado;

Além de definir as informações que devem estar no *Dashboard,* é necessário também definir o design que será apresentado ao cliente final. Para isso, foram feitos alguns esboços com o objetivo de delimitar em qual parte da página ficaria cada gráfico, tabela, filtros e todas as informações que devem ser apresentadas.

Como o cliente é fascinado por aplicações mobile, além de desenvolver o layout para usuário de computadores, foi produzido um design para celulares e tablets.

5.2: Modelagem Banco de Dados

Com o design do Dashboard definido, passamos para a modelagem do banco de dados, no qual todas as informações serão armazenadas. Para tanto, foi utilizada a metodologia *Star Schema* (ou Esquema em Estrela) de acordo com o que foi apresentado na seção 4.1: deste documento.

Utilizando o software Oracle SQL Developer Data Modeler, o seguinte banco de dados foi modelado:

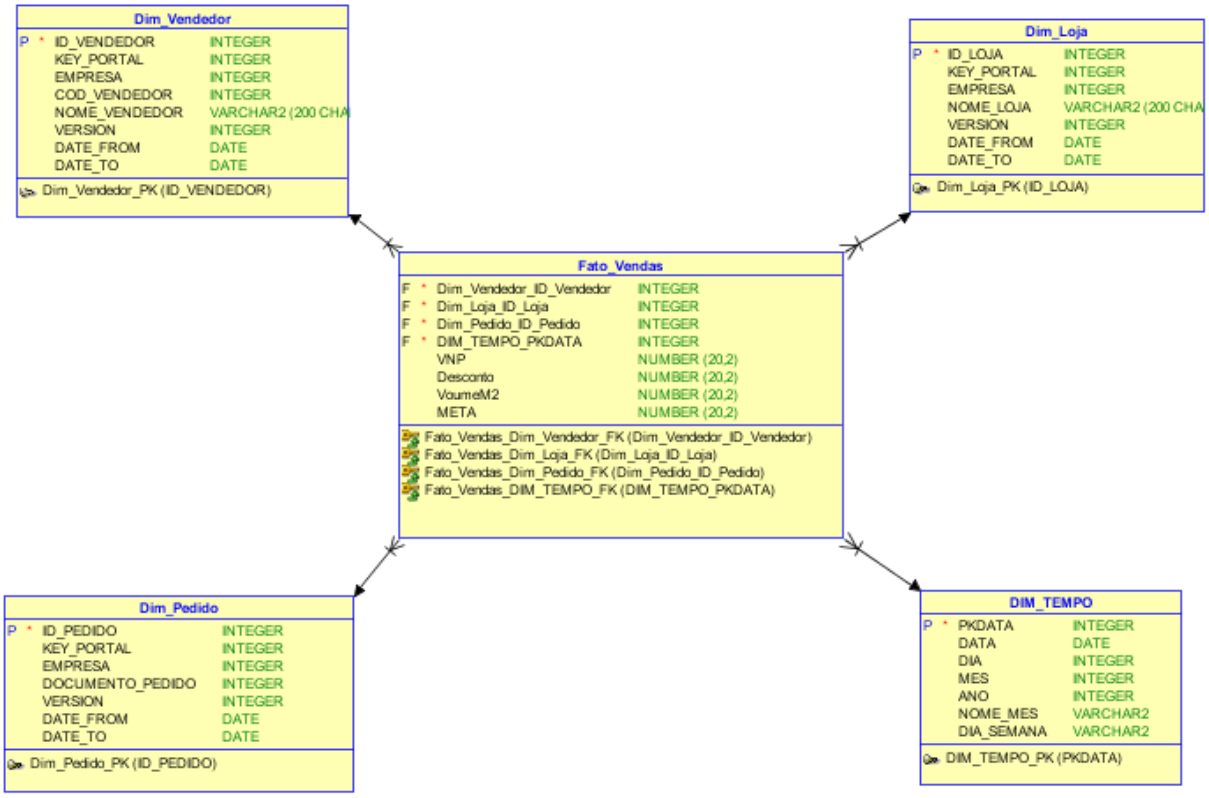

Figura 4 - Modelagem do Banco de Dados

Dessa forma, as seguintes dimensões e tabela fato foram definidas, como exibida na Figura 4 - Modelagem do Banco de Dados:

A dimensão Tempo foi criada contendo as seguintes colunas: PKDATA, DATA, DIA, MÊS, ANO, NOME_MES e DIA_SEMANA.

A Dimensão Loja foi modelada com as seguintes colunas: ID_LOJA, KEY_PORTAL, EMPRESA, NOME_LOJA, VERSION, DATE_FROM e DATE_TO.

A Dimensão Vendedor foi criada contendo as colunas: ID_VENDEDOR, KEY_PORTAL, EMPRESA, COD_VENDEDOR, NOME_VENDEDOR, VERSION, DATE_FROM e DATE_TO.

A Dimensão Pedido foi modelada para possuir as colunas: ID_PEDIDO, KEY_PORTAL, EMPRESA, DOCUMENTO_PEDIDO, VERSION, DATE_FROM e DATE_TO.

A tabela Fato Vendas contém as colunas: DIM_LOJA_ID_LOJA, DIM_VENDEDOR_ID_VENDEDOR, DIM_PEDIDO_ID_PEDIDO, DATA_VENDA, VNP, DESCONTO, VOLUMEM2 e META.

5.3: ETL (Extract Transform Load)

Para cada dimensão modelada anteriormente e para a tabela fato, foi necessário desenvolver uma transformação (*transformation* em inglês) utilizando a ferramenta Spoon do software Pentaho Data Integration. Essa ferramenta é responsável por extrair os dados vindos de uma única tabela com informação de todos os pedidos e carregar essas informações em suas devidas dimensões, para que assim, a consulta dos dados aconteça de forma mais rápida.

Nas transformações responsáveis pela população das tabelas Dimensão, como mostra a Figura 5, foram utilizados os steps “Table input” e “Dimension lookup/update”, responsáveis por extrair os dados da tabela de vendas e atualizar a tabela dimensão, respectivamente.

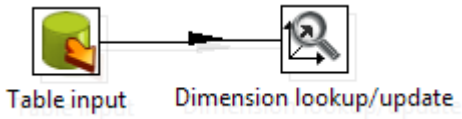

Figura 5 - Transformação Dimensões

Para configurar o step “Table input”, mostrado na Figura 6, é necessário definir o campo “connection” com o nome do banco de dados de onde as informações serão extraídas, neste caso “PB_DW”. Feito isso, é desenvolvido um código SQL para definir qual tabela do banco de dados e quais informações devem ser extraídas dessa tabela. No exemplo mostrado na Figura 6, são extraídas as informações contidas nas colunas “Key_Portal”, “Empresa” e “Nome_Loja” da tabela “STG_DADOS_VENDAS”.

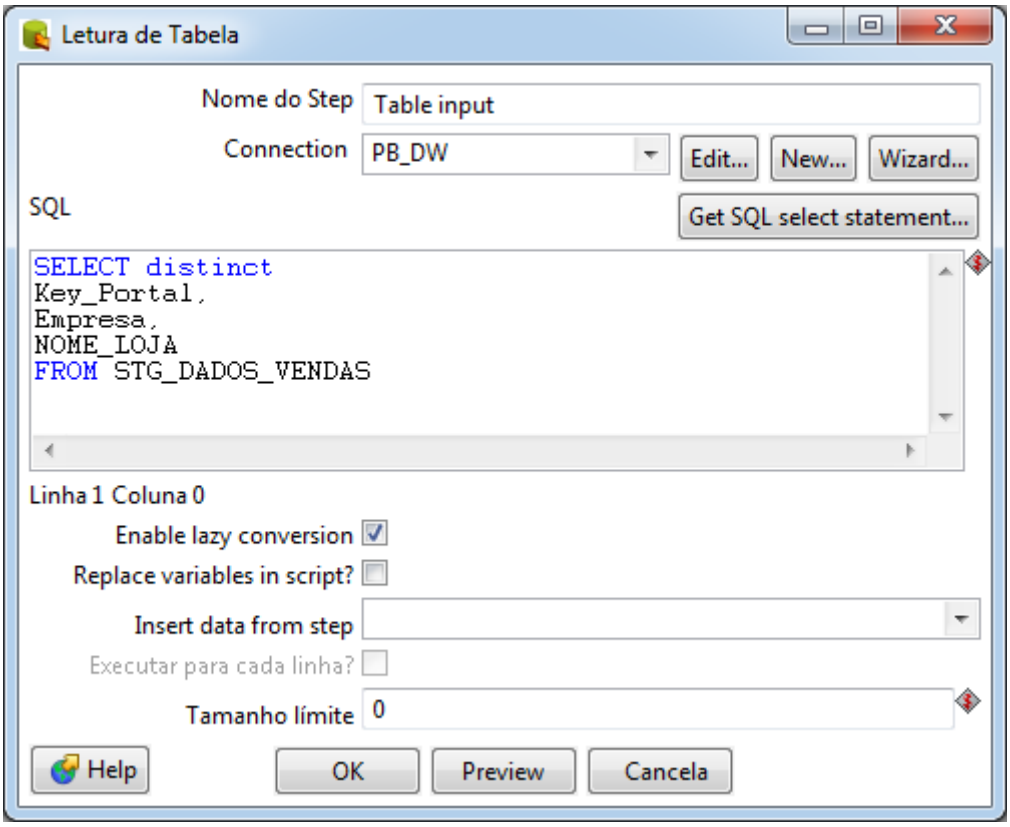

Figura 6 - Step Table Input da Transformação da Dimensão Loja

Para realizar a atualização das informações contidas na Dimensão Loja, foi utilizado o *step* “Dimension Lookup/Update”, o qual utiliza as informações fornecidas pelo *step* anterior para atualizar a dimensão com as informações novas, caso alguma informação seja alterada.

Os campos a serem definidos no *step* “Dimension lookup/update” são mostrados na Figura 7. É preciso definir em qual banco de dados está a tabela que será atualizada através do campo “Connection”, a tabela que será atualizada deve ser definida no campo “Target table”. Na aba “Key” são definidas quais colunas vindas do *step* anterior devem ser comparadas com as colunas presentes na tabela dimensão. Na aba “Fields”, mostrado na Figura 8, define-se as colunas que devem ser comparadas, que caso haja diferença, deve-se criar uma nova versão dessa dimensão. No campo “Technical key field” define-se o nome da coluna que será criada para identificação de cada linha da dimensão, este campo será utilizado como chave primária nas conexões com a tabela fato. O campo “version” define o nome da coluna que identificará a versão do membro que está sendo identificado

naquela linha. Os campos “date_from” e “date_to” são utilizados para definir qual o período no tempo que cada linha deve ser utilizada caso haja mais de uma versão da informação.

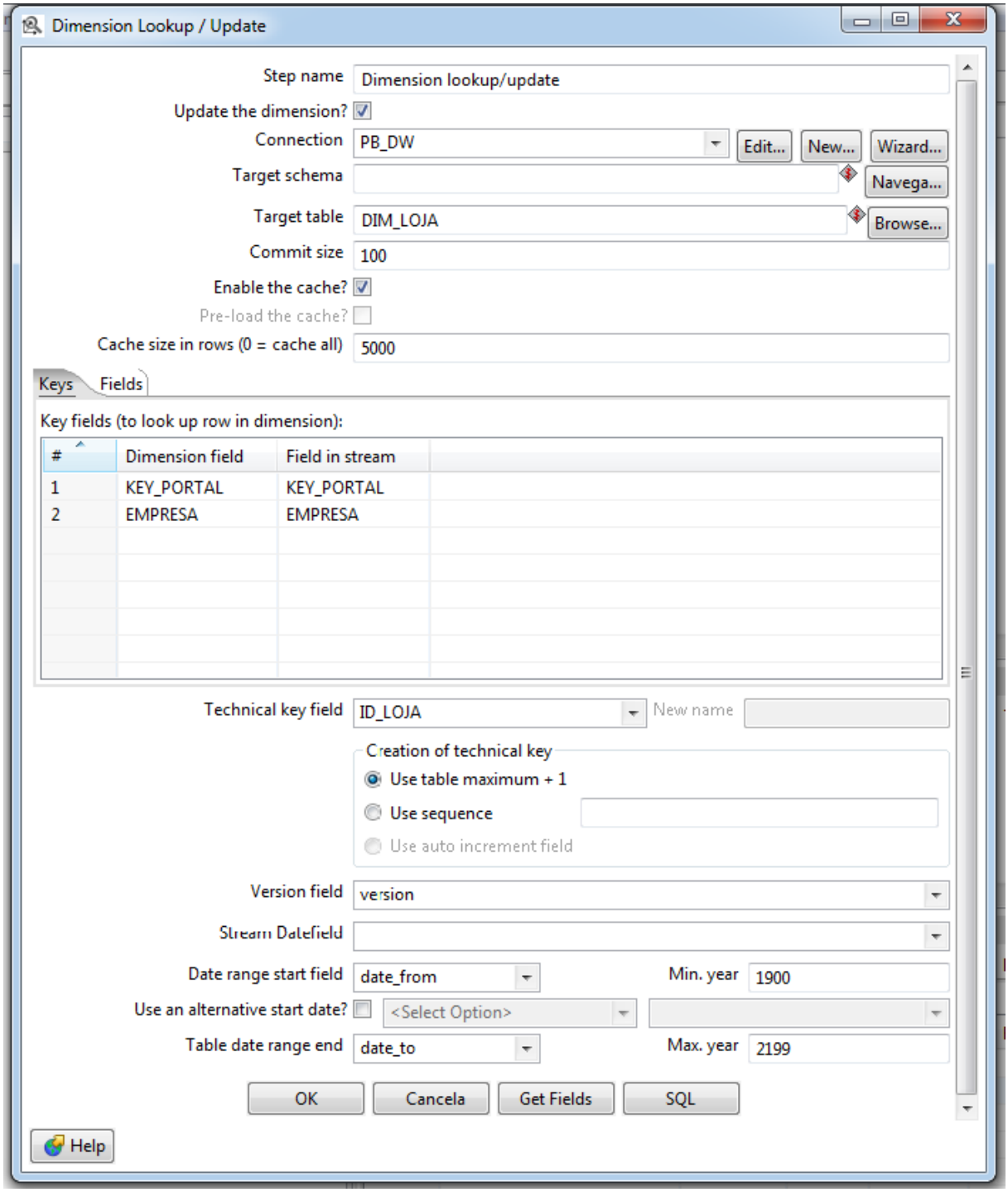

Figura 7 - Step Dimension Lookup/Update da Transformação da Dimensão Loja

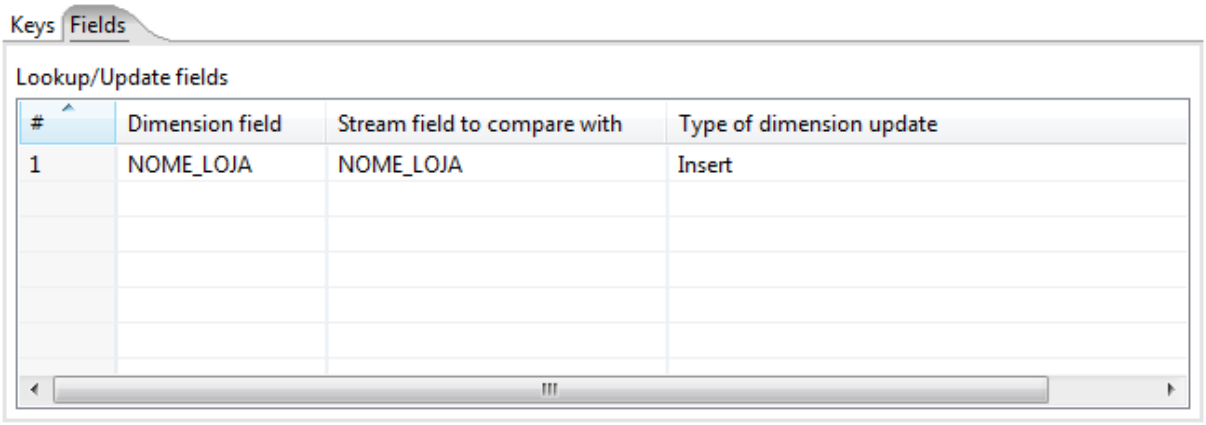

Figura 8 - Aba Fields do Step Dimension Lookup/Update

Por exemplo, a loja definida como empresa 1 fechou e agora, outra loja foi aberta e será identificada como empresa 1. A loja fechada permanecerá na Dimensão Loja como sendo empresa 1, porém com a informação “date_to” até a data de quando foi fechada, já a nova loja será identificada com empresa 1 e terá seu campo “date_from” com a data de quando foi cadastrada no banco de dados. Com isso, o sistema é capaz de reconhecer na tabela fato a loja definida como empresa 1 dependendo da data de quando o pedido foi realizado.

Na transformação utilizada para popular a tabela fato, como mostra a Figura 9, foram utilizados os steps “Table input”, “Database Lookup”, “Select Values” e “Table Output”. O primeiro step é responsável por extrair os dados da tabela de vendas. O step “Database Lookup” faz as conexões com as tabelas dimensões. Já os steps “Select Values” e “Table Output” são responsáveis por organizar as informações e escrever as informações na tabela fato, respectivamente.

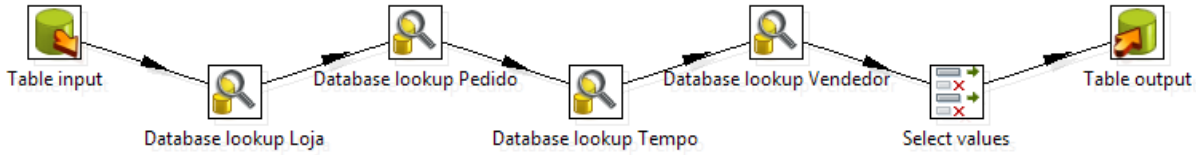

Figura 9 – Transformação Tabela Fato

Como explicado anteriormente, o step “Table Input” utiliza um código SQL para realizar a busca e extração dos dados de uma determinada tabela do banco de dados. Nesse caso, são extraídas as informações dos pedidos contidas na tabela “STG_DADOS_VENDAS”.

É utilizado um step “Database Lookup” para cada dimensão existente no projeto (Loja, Pedido, Tempo e Vendedor). Esses steps são responsáveis por realizar consultas nas Tabelas Dimensões e identificar qual dimensão está sendo referida e trazer apenas um valor inteiro para a tabela fato. A composição da tabela fato com informações do tipo inteiro resulta em maior rapidez de pesquisa.

Para isso, como mostra na Figura 10, alguns campos devem ser definidos com o propósito que a pesquisa seja feita de forma correta e eficiente. Através do campo “Tabela Lookup” é definida qual tabela dimensão deve ser utilizada para a realização da pesquisa. Em seguida, são definidos alguns campos para que sejam realizadas as comparações necessárias. E então, é definido o campo que deve ser inserido na tabela fato como FK, conectando assim a tabela fato à tabela dimensão.

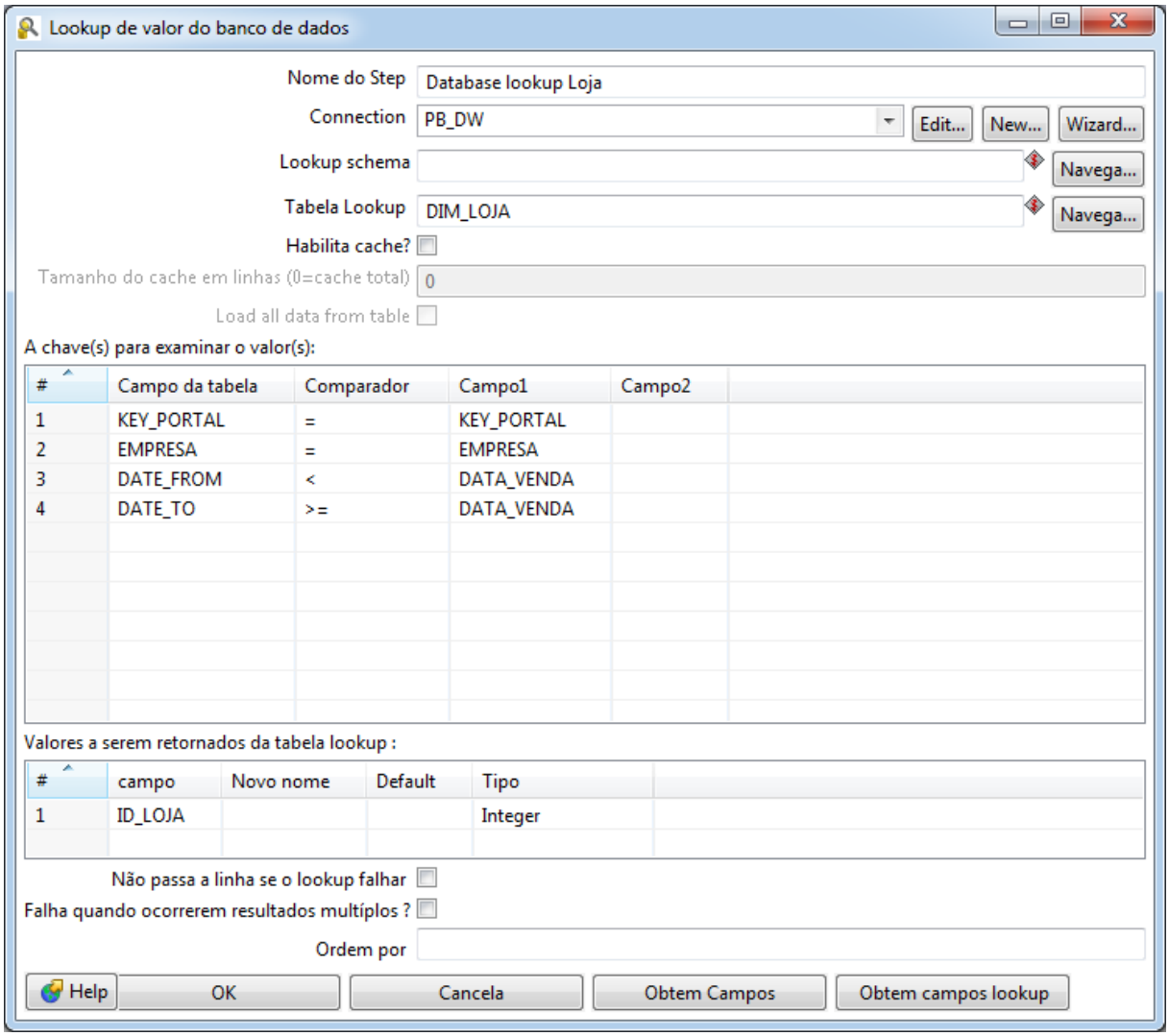

Figura 10 - Configurações Step Database Lookup

Após todas as pesquisas serem feitas em suas respectivas dimensões, é utilizado o *step* “Select Value” com o objetivo de definir quais serão as informações escritas na tabela fato. E então se utiliza o *step* “Table Output” para realizar a escrita das informações na tabela fato.

Além dessa transformação, outra foi desenvolvida a fim de adicionar os valores das metas das lojas à tabela fato, visto que, as informações das metas são extraídas de outra tabela, como mostra a Figura 11. Essa utiliza os mesmos *steps* da transformação explicada anteriormente e é responsável por pegar as informações da tabela de metas e fazer a pesquisa nas dimensões Loja e Tempo, para que na tabela fato contenham apenas informações do tipo inteiro e numérico.

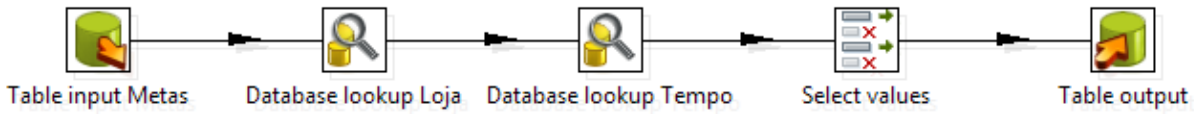

Figura 11 - Transformação Metas Tabela Fato

Com a finalização dessa transformação, a tabela fato apresenta informações apenas do tipo inteiro e numérico, como mostra a Figura 12. Dessa forma, a tabela fica menos pesada e facilita uma futura consulta aos dados, tornando as pesquisas que serão realizadas pelo sistema de BI muito mais rápidas e eficientes.

DIM_LOJA_ID_LOJA	DIM_VENDEDOR_ID_VENDEDOR	DIM_PEDIDO_ID_PEDIDO	DATA_VENDA	VNP	DESCONTO	VOUMEM2	META
3	139	13363	20160507	44,05	4,91	0	(null)
3	132	15599	20160507	2106,566	229,05	19,84	(null)
3	23	18300	20160507	456,53	50,72	2,5	(null)
4	64	13707	20160507	274,00399...	30,8	6,35999...	(null)
4	14	11425	20160507	9526,781	832,92	90,09	(null)
5	100	456	20160507	720	73,5	0	(null)
5	9	616	20160506	3721,763	0	28,47	(null)
4	14	6960	20160506	1098,998	0,1	20,02	(null)
5	9	617	20160506	1090,005	23,25	7,15	(null)
5	100	455	20160506	82,786999...	0,01	1,43	(null)
5	55	13945	20160506	5571,7090...	0	52,91	(null)
6	75	713	20160506	9933,1120...	747,6500...	89,7299...	(null)
5	111	7581	20160506	557,27100...	0	4,29	(null)
7	37	916	20160506	126,57600...	0	1,44	(null)
7	107	12561	20160506	502,66	68,54	0	(null)
7	107	14820	20160506	4403,5160...	777,09	54,74	(null)

Figura 12 - Tabela Fato

5.4: Conectando Banco de Dados com *BI Server*

Após popular todas as tabelas do banco de dados é estabelecida a conexão entre o *BI Server* e o banco de dados para que o servidor tenha acesso a todas as informações das vendas.

Para realizar essa conexão, é necessário adicionar o *Driver* do banco de dados à pasta específica nos arquivos do servidor. Feito isso, inicia-se o servidor e então se cria uma nova conexão de dados colocando as informações do banco de dados como IP, porta, usuário, etc.

5.5: Cubo de Dados

Para a criação do cubo de dados foi utilizada a ferramenta *Schema Workbench* que faz parte da suíte Pentaho. Essa ferramenta auxilia a criação do cubo OLAP. Foram criadas todas as dimensões do cubo: Dimensões Tempo, Loja, Vendedor e Pedido, e feitas todas as conexões entre as tabelas dimensão criadas no banco de dados e a tabela fato, também criada no banco de dados.

Para realizar essa conexão entre a Dimensão e a Tabela Fato, é necessário informar qual coluna, na tabela Fato, contém a Foreign Key (ou Chave Estrangeira) que informa qual linha da Dimensão especificada possui as informações da dimensão. Por exemplo, na tabela fato, temos na coluna DIM_LOJA_ID_LOJA apenas o inteiro 2, isso nos conecta com a Dimensão Loja, que nos informa que o ID_LOJA 2 significa KEY_PORTAL = 2926, EMPRESA = 5, NOME_LOJA = LOJA B.

5.5.1: Conectando o *Schema Workbench* com o Banco de Dados

Antes de iniciar a criação do cubo de dados, é necessário conectar o software *Schema Workbench* com o banco de dados. Fazendo isso, o software de criação do cubo terá acesso às tabelas do banco dados bem como a todas as colunas das tabelas.

Para conectar o programa com o banco de dados, é necessário adicionar o *Driver* do banco de dados à pasta especifica nos arquivos do software. Deve-se então realizar a conexão com o banco de dados do mesmo modo como foi realizado anteriormente para conectar o servidor ao BD.

5.5.2: O Cubo

Nesse projeto foi criado o cubo “PB Vendas Metas” como mostra a Figura 13. Este cubo é associado à tabela fato FATO_VENDAS_METAS. As seguintes Dimensões estão contidas nesse cubo:

Loja – Associada à tabela DIM_LOJA e possuindo os níveis Portal, Empresa e Nome Loja;

Vendedor - Associada à tabela DIM_VENDEDOR e contendo os níveis Portal, Empresa, Código Vendedor e Nome Vendedor;

Pedido - Associada à tabela DIM_PEDIDO e contendo os níveis Portal, Empresa e Documento Pedido;

Tempo – Associada à tabela DIM_TEMPO e contendo os níveis Ano, Mês e Dia.

A dimensão Tempo foi criada fora do cubo e adicionada ao cubo como uma "Dimension Usage". Desse modo, caso sejam criados outros cubos, esses podem utilizar a mesma dimensão tempo, evitando, assim, uma dimensão tempo para cada cubo criado.

Além das dimensões citadas anteriormente, nesse projeto, foram criadas as seguintes medidas: VNP, Desconto, Número de Pedidos, Volume M2, Meta e Número de Vendedores. Essas medidas são informações obtidas diretamente da tabela fato, sem requerer nenhum tipo de expressão para cálculo.

Entretanto, foi necessário criar algumas Medidas Calculadas, que são medidas resultantes de um cálculo. Nesse projeto, foram criadas as seguintes medidas calculadas: Porcentagem de Desconto, Ticket Médio e Preço Médio m². Por exemplo, para calcular a medida "Porcentagem de Desconto" utilizou-se a expressão:

([Measures].[Valor Desconto])/([Measures].[VNP] + [Measures].[Valor Desconto])

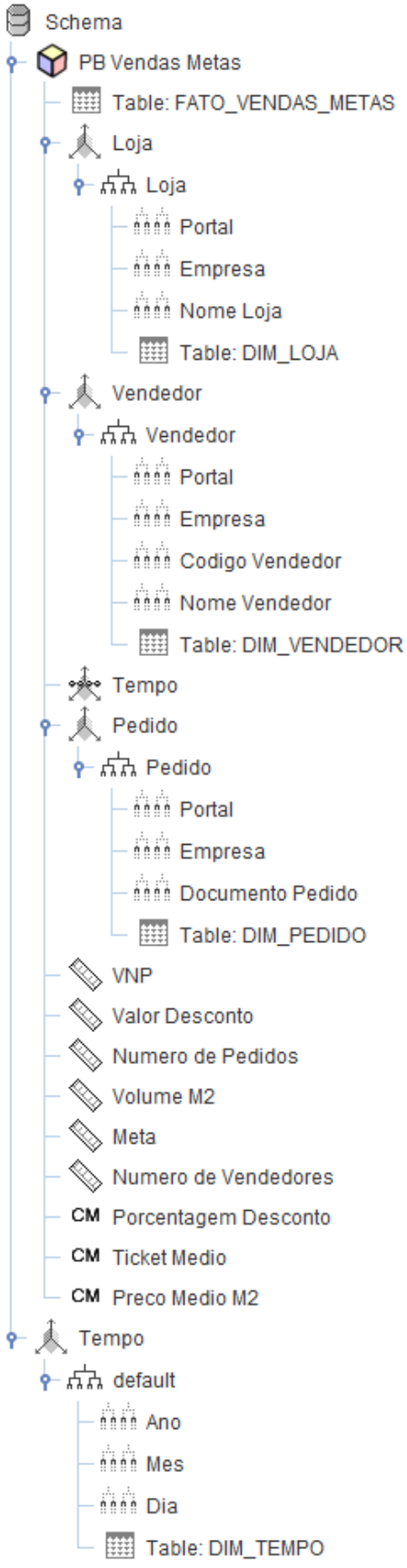

Figura 13 - Cubo de Dados

5.5.3: Publicando o Cubo

Após desenvolver todas as dimensões e medidas do cubo, é necessário conectá-lo com o *BI Server*. Devido ao software de criação do cubo, *Schema Workbench*, fazer parte da mesma suíte do *BI Server*, o processo de publicação do cubo não exige muito trabalho.

Para realizar a publicação do cubo, deve-se identificar o *DataSource* o qual o cubo irá obter as informações. O nome desse *DataSource* foi definido na conexão entre o server e o BD como foi apresentado na seção 5.4: deste documento.

5.6: Criação do *Dashboard*

Com o cubo de dados publicado no servidor e esse conectado com o banco de dados no qual estão todas as informações das vendas, inicia-se a criação do *Dashboard*. O Dashboard é um painel que irá apresentar as informações mais relevantes para o cliente final, com o objetivo de auxiliá-lo nas tomadas de decisão.

Com o intuito de manter uma organização no projeto e auxiliar o desenvolvimento do produto, essa fase de criação do painel foi dividida nas seguintes partes: Layout, Componentes, Filtros (Variáveis), Datasources, Mobile, Melhorias (CSS e JS) e Exportação para Excel.

5.6.1: Layout

Com o layout já definido, essa etapa consiste em criar o layout diretamente no servidor. Para esse fim, o *BI Server* possui uma interface chamada *Layout*, como mostra a Figura 14 e em que se define como serão dispostos os componentes no painel.

Utilizando o esboço criado no início do projeto, foi definido um *Dashboard* com quatro linhas, sendo elas:

Cabeçalho (Header) - Barra de Navegação: essa linha será utilizada para exibir o logo da empresa como também os seletores dos filtros que serão aplicados no painel.

Primeira Linha (r1) - Gráficos: essa linha será dividida em duas colunas. A coluna da esquerda contém uma tabela na qual são mostradas informações de cada loja. Já na segunda coluna, é exibido um gráfico com as informações de vendas da loja selecionada.

Segunda Linha (r2) – Bullets: essa linha possui quatro colunas, as quais apresentam algumas informações relevantes da loja selecionada.

Terceira Linha (r3) – Tabela: essa linha apresenta uma tabela contendo informações sobre os vendedores. Sendo assim, possui apenas uma coluna, a qual contém a tabela, e essa sim contém algumas colunas.

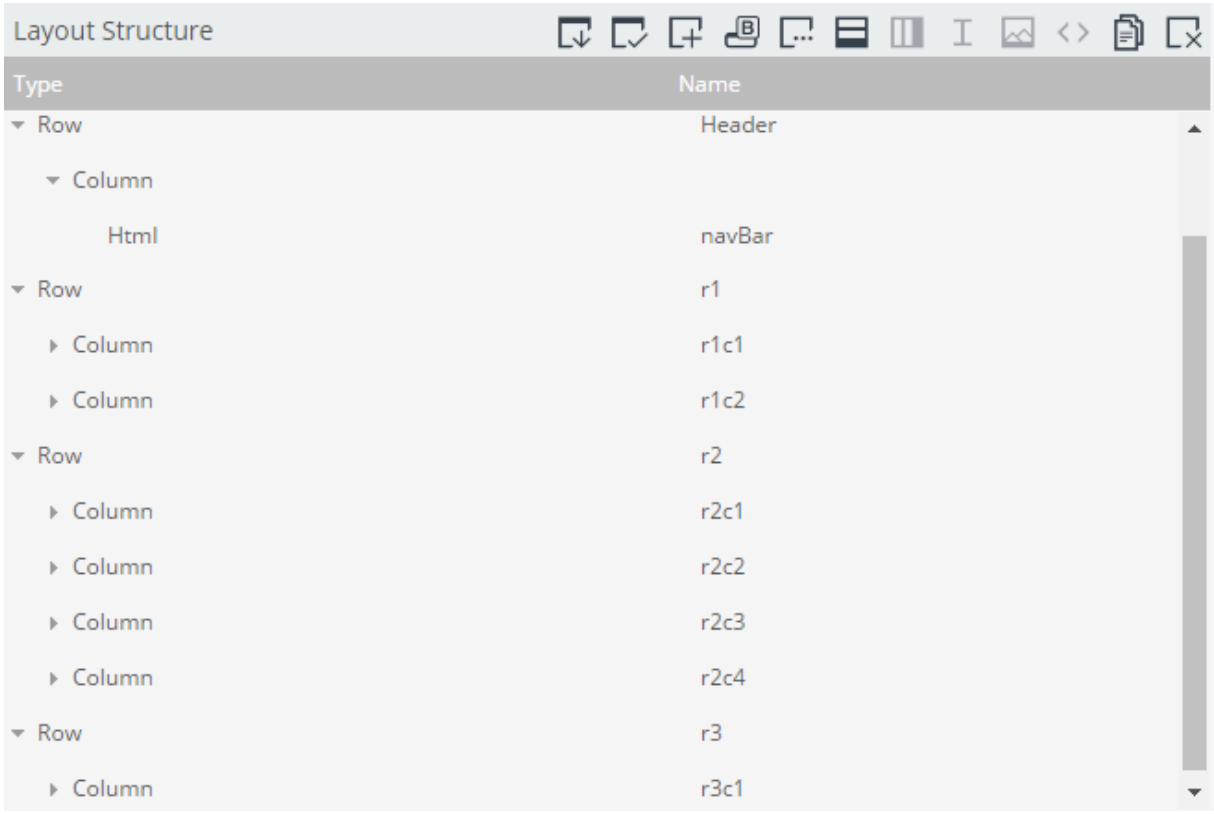

Figura 14 - Estrutura do Layout

5.6.2: Componentes

Definidos os espaços que cada componente ocupa, na área de layout, iniciou-se então a criação dos componentes que fazem parte do *Dashboard*. Esses componentes incluem: seletores, tabelas, gráficos, entre outros itens, conforme Figura 15.

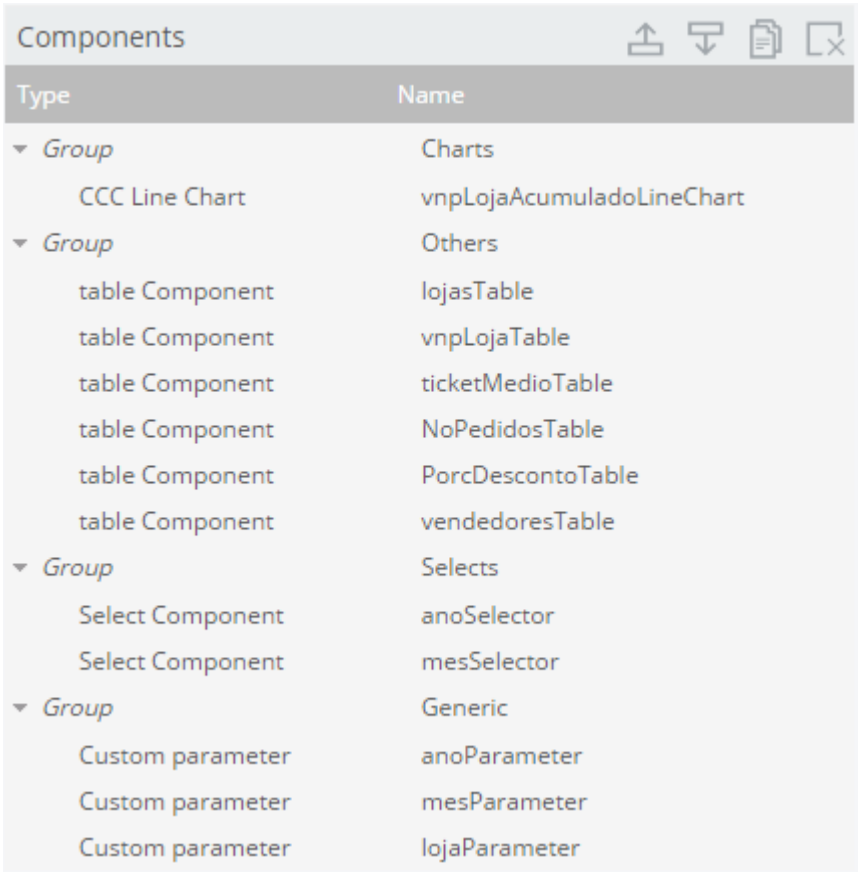

Figura 15 - Componentes

5.6.2.1: Tabela de Lojas

Foi definido que, nesta tabela, devem ser informados o valor de venda, a meta e qual foi o percentual de atingimento da meta para cada loja. Logo, essa é uma tabela que contém quatro colunas e cada linha representa uma loja.

Para criar essa tabela, adicionou-se um novo componente tabela e ajustou-se suas configurações para que estejam de acordo com as informações que são apresentadas.

Na Figura 16, observa-se algumas propriedades que foram ajustadas para que as informações desejadas sejam apresentadas. Por exemplo, a propriedade “Datasource” foi definida como “vendasLojaQuery”, que é o elemento responsável por buscar as informações sobre as lojas no cubo de dados (esse elemento será melhor explicado na seção 5.6.4:; já a propriedade “HtmlObject” é definida como

“r1c1”, que identifica o local no qual a tabela será apresentada no Dashboard, como foi explicado na seção 5.6.1:.

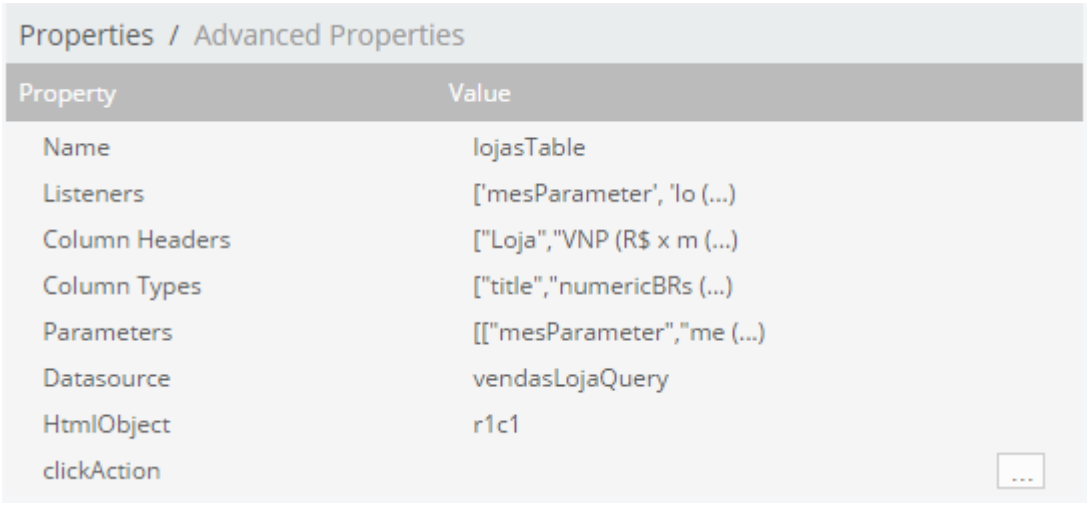

Figura 16 - Propriedades da Tabela de Lojas

5.6.2.2: Gráfico VNP

Este gráfico tem como objetivo mostrar visualmente o desempenho da loja durante um mês em relação ao seu VNP. Desse modo, definiu-se que esse gráfico deve apresentar em seu eixo X os dias do mês e em seu eixo Y, o valor VNP acumulado realizado na loja.

Criou-se então um novo componente de gráfico de linha (*Line Chart*). Dessa forma, algumas alterações em suas configurações foram realizadas para que esse gráfico exibisse corretamente as informações desejadas, como mostra a Figura 17. Do mesmo modo que a tabela criada anteriormente, deve-se definir as propriedades “Datasource” e “HtmlObject” a fim de informar quais informações devem ser apresentadas e o local no Dashboard no qual o gráfico deve aparecer, respectivamente.

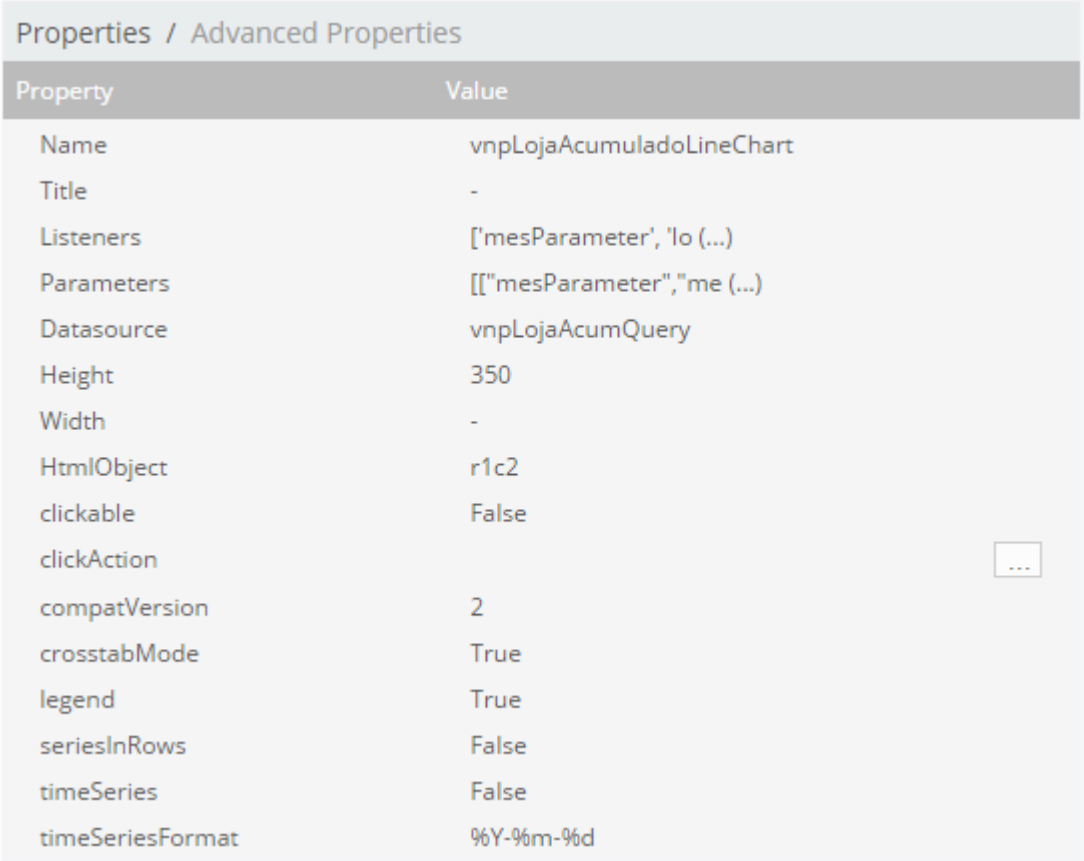

Figura 17 – Propriedades do Gráfico de Venda Acumulada

5.6.2.3: Bullet Points

Bullet Points são utilizados para chamar atenção às informações relevantes. Sendo assim, esse *Dashboard* apresenta quatro *bullets*, os quais informam o VNP, o Ticket Médio, o Número de Pedidos e a Porcentagem de Desconto da loja.

Com o objetivo de criar esses componentes, foi utilizada uma tabela para cada informação a ser apresentada. Dessa forma, quatro componentes 'tabela' foram criados, e assim como os componentes criados anteriormente, algumas propriedades tiveram que ser alteradas para que as informações fossem apresentadas do modo desejado.

5.6.2.4: Tabela de Vendedores

A fim de indicar informações referentes a cada vendedor, foi criada uma tabela com as seguintes colunas: Nome do Vendedor, VNP, Ticket Médio, Número

de Pedidos e Porcentagem de Desconto. Desse modo, cada linha da tabela referencia um vendedor. Para que as informações sejam exibidas da forma desejada, foi necessário alterar algumas propriedades, assim como foi realizado nos componentes anteriores.

5.6.3: Filtros (Variáveis)

Para que o cliente consiga interagir com o *Dashboard,* foram criados filtros (ou variáveis). Desse modo, o usuário tem a possibilidade de filtrar as informações que aparecem no painel, definindo o mês e ano desejado, bem como também uma loja específica.

Para criar os filtros, na área de componentes, adicionaram-se componentes “Simple Parameter”, como mostrado na Figura 18.

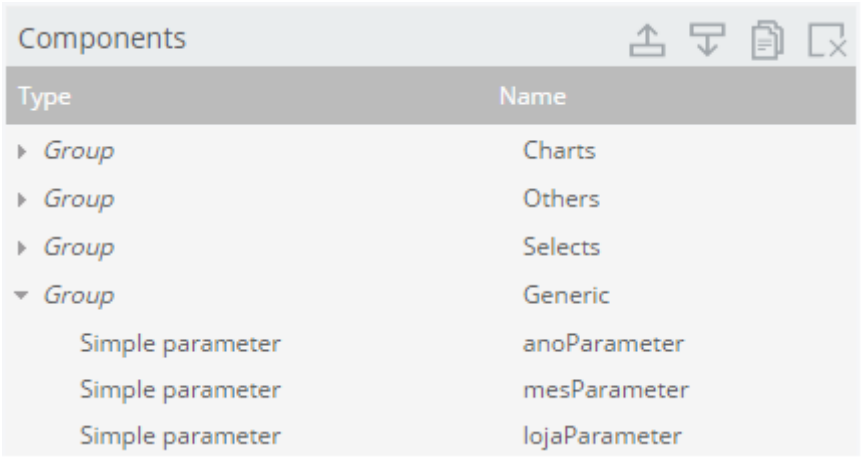

Figura 18 - Componentes de Parâmetros

A Figura 19 exibe as propriedades que devem ser definidas quando um parâmetro simples é criado. Nesse caso, “Name” define o nome do parâmetro como “anoParameter” e “Property Value” define o valor padrão do parâmetro como ano de 2016.

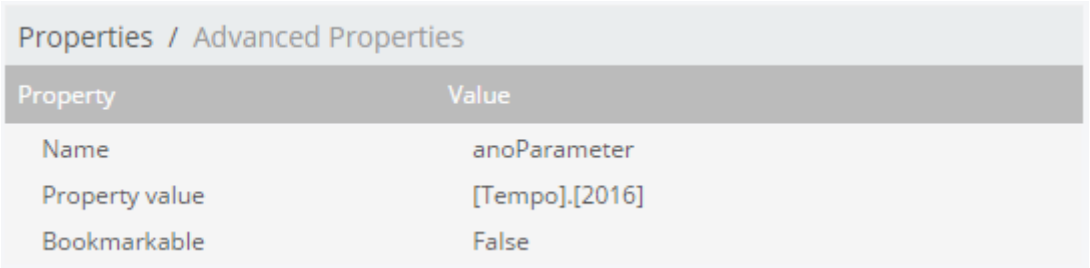

Figura 19 - Propriedades de Parâmetros

5.6.4: Datasources

Criados os componentes os quais exibem as informações desejadas, foram desenvolvidos os códigos que realizam as consultas no cubo OLAP e trazem os dados necessários para serem exibidos.

Desse modo, foi necessário desenvolver um código MDX para cada componente citado anteriormente, visto que cada gráfico ou tabela traz informações diferentes.

Para isso, criou-se na área de *Datasource*, elementos “mdx over mondrianJndi”, os quais são responsáveis por realizar as buscas de informações no cubo de dados e enviar aos componentes criados na seção 5.6.2:.

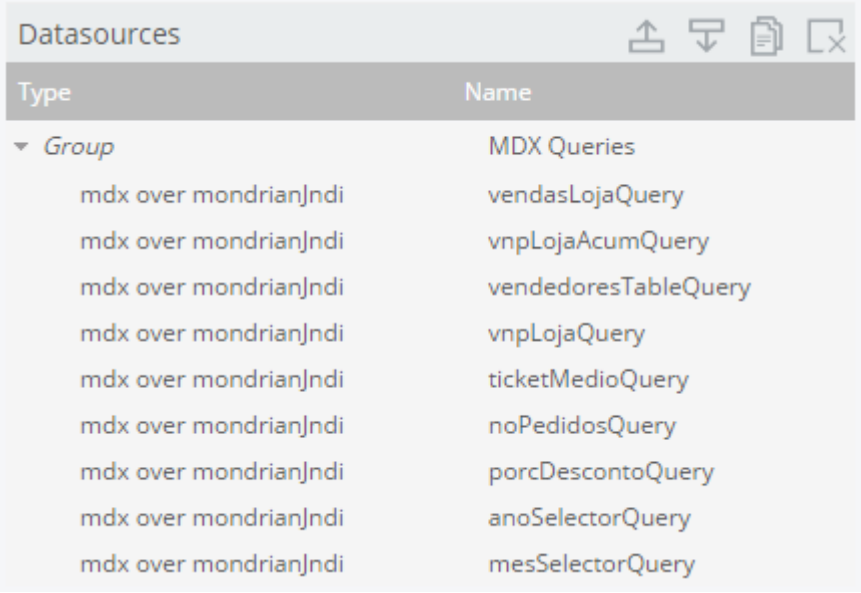

Figura 20 - Elementos de Conexão com o Cubo de Dados

Cada elemento presente na Figura 20 é responsável por obter os dados desejados para serem apresentados em cada elemento. Sendo assim, foi necessário alterar algumas propriedades desses elementos como mostra a Figura 21.

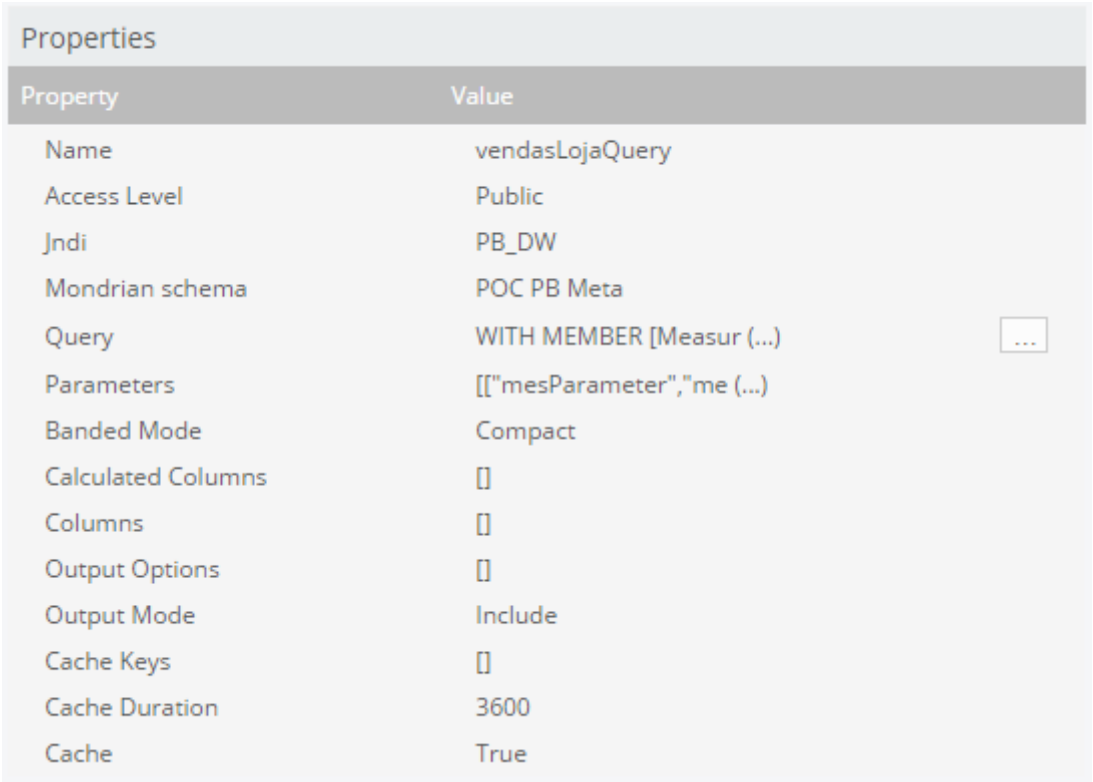

Figura 21 - Propriedades do Datasource

Todos os *Datasources* utilizados nesse projeto possuem os valores das propriedades “Jndi” e “Mondrian schema” como “PB_DW” e “POC PB Meta” respectivamente, já que esses sempre se referem ao mesmo banco de dados e *Schema*.

Contudo, como cada *Datasource* deve trazer informações diferentes, cada elemento possui uma “Query” específica. As *Queries* desenvolvidas nesse projeto estão apresentadas no apêndice, ao final deste documento.

5.6.5: Mobile

Um dos objetivos desse projeto é possibilitar o acesso às informações do BI por meio de equipamentos mobile (celulares e tablets). Desse modo, foi necessário

definir um layout do Dashboard para a apresentação nesses aparelhos, uma vez que suas telas são menores que às de um computador. Além disso, também foi preciso desenvolver um código em JavaScript para que o Dashboard se adequasse ao tamanho da tela do dispositivo.

5.6.5.1: Layout

Devido às telas de aparelhos mobile serem menores que às de computadores e televisões e apresentarem formatos diferentes, foi necessário definir layouts diferentes como o intuito de obter uma boa visualização em todos os dispositivos.

Para que essa adaptação fosse realizada, adicionou-se componentes chamados *Bootstrap Panels* na interface *Layout*, a mesma citada na seção 5.6.1:. Dessa forma, cada componente criado anteriormente deve estar referenciado a um *Bootstrap Panel* específico, como mostra a Figura 22.

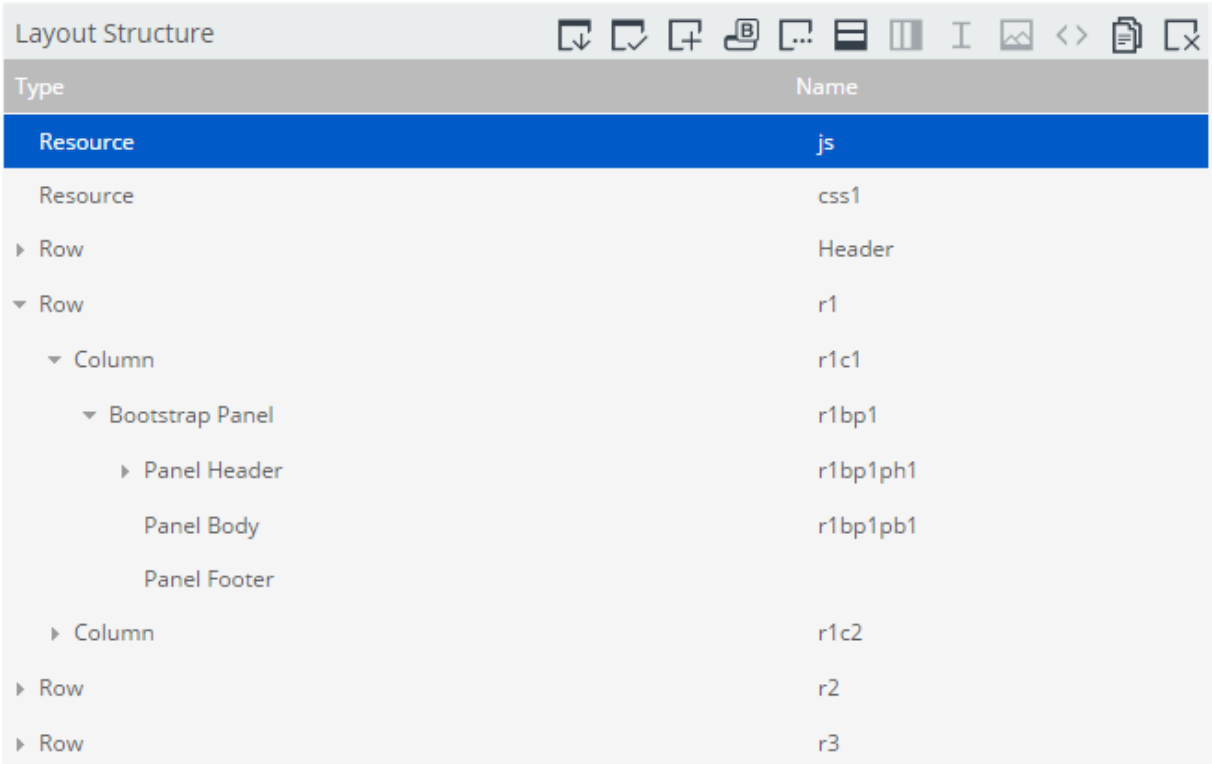

Figura 22 - Componente Bootstrap Panel

5.6.5.2: JavaScript

Além de adicionar os painéis *bootstrap* como citado anteriormente, foi necessário desenvolver um código em JavaScript de modo que os componentes gráficos se adaptem ao tamanho da tela do dispositivo que está sendo utilizado. Esse código também entra em ação caso a janela do navegador, em um computador, altere de tamanho. Ou seja, tal código tem como objetivo deixar o painel com a largura fixa do dispositivo que está sendo utilizado.

O código em JavaScript adicionado na propriedade "*Pre* execution", situada em *Advanced Properties* de cada componente gráfico criado anteriormente encontra-se no apêndice deste documento.

5.6.6: Melhorias (CSS e JS)

Para que o Dashboard tenha uma melhor aparência e agrade o cliente, algumas alterações visuais foram realizadas. Para tal, foi desenvolvido um código em CSS (*Cascade Style Sheets*) e também em JavaScript.

5.6.6.1: Código CSS

CSS é uma linguagem de programação que possui diversas instruções para definir como um navegador de internet deve apresentar um determinado *website*, neste caso o Dashboard desenvolvido. [7]

Com o intuito de que o Dashboard apresente uma aparência mais amigável, foi desenvolvido um código CSS que se encontra no final deste documento, na área de apêndice.

5.6.6.2: Código JS para Exibição de Valores Numéricos

Por não ser um software brasileiro, o Pentaho utiliza como separador de milhares numérico o caractere "," e o separador de casas decimais ".". Por exemplo, o valor de dez mil reais e ciquenta centavos é representado como "R$10,000.50". De forma a adaptá-lo ao sistema brasileiro, foi desenvolvido um código utilizando

linguagem JavaScript, o qual apresenta o valor citado anteriormente como “R$10.000,50”.

Além disso, o código desenvolvido é responsável por definir o formato em que o número será exibido, adicionando caracteres como “R$” e “%” quando necessário, bem como definindo o número de casas decimais que deve ser exibido.

O código foi desenvolvido em JavaScript e está apresentado no final deste documento, na área de apêndice.

5.6.6.3: Código JS para Interação com Tabela de Lojas

Com o propósito de deixar o Dashboard ainda mais interativo, fez-se com que a seleção da loja, responsável por filtrar as informações do painel por loja, fosse realizada clicando na tabela de lojas e não mais através de um objeto de seleção como os do mês e ano.

Para que isso seja possível, foi necessário desenvolver um código em JavaScript para reconhecer o clique do usuário e poder realizar as alterações necessárias no Dashboard. Além disso, o código é responsável por identificar qual loja está sendo filtrada e, juntamente com um código em CSS, destacá-la na tabela de lojas. O código foi desenvolvido e adicionado na propriedade “Post Execution” do componente da tabela de lojas e está apresentado ao final deste documento, na área de apêndice.

5.6.6.4: Código JS para Iniciar Dashboard com Mês Atual

Para que o Dashboard seja iniciado sempre no mês atual, foi desenvolvido um código em JS com o objetivo de definir o valor padrão dos parâmetros ano e mês com os valores atuais. Para isso, os componentes “Simple parameter” foram substituídos por “Custom Parameter” e um código foi desenvolvido para cada parâmetro e adicionado na propriedade “Javascript Code”, esse código é apresentado ao final deste documento, na área de apêndice.

5.6.7: Exportação para Excel

Além de exibir as informações necessárias no painel, uma funcionalidade de Dashboards de BI muito comum é a exportação dos dados apresentados em formato de planilha (xls). Para isso, foram adicionados botões em cada tabela/gráfico responsáveis por exportar as informações apresentadas em cada componente em formato “xls”.

Para esse fim, na área de layout foram adicionados os lugares os quais os botões de exportação de cada gráfico/tabela seriam adicionados. Definidos os lugares, criaram-se os componentes responsáveis pela exportação dos dados na área de componentes, a mesma na qual são criadas as tabelas e gráficos, como mostra a Figura 23.

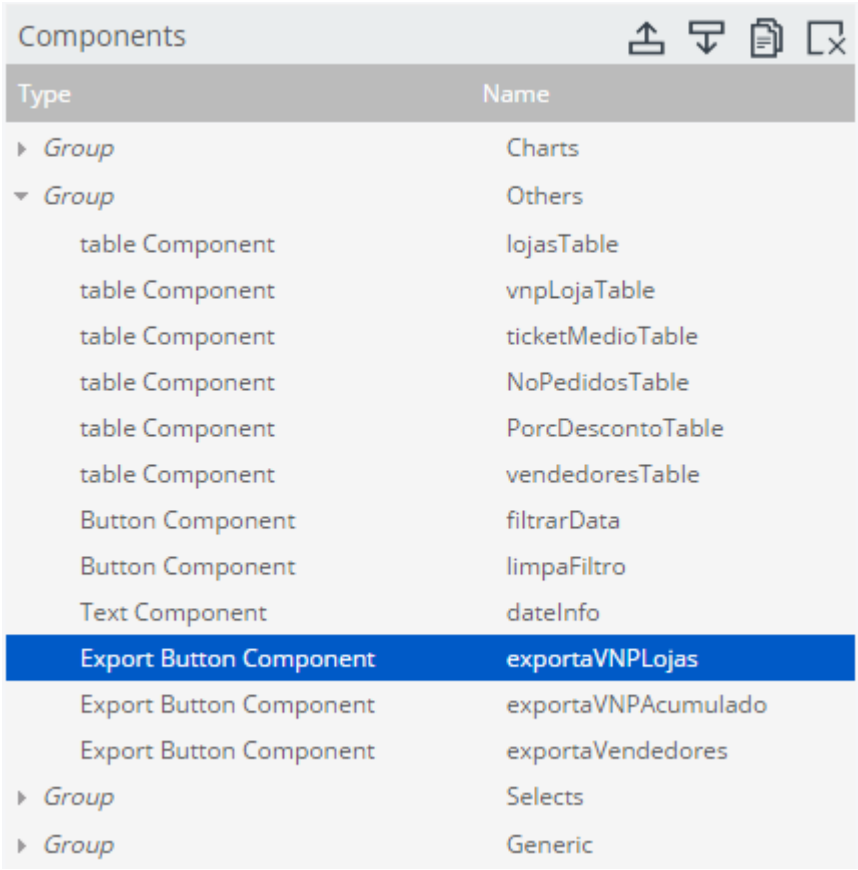

Figura 23 - Componente de Exportação XLS

Como mostra a Figura 24 algumas propriedades foram definidas para determinar onde são apresentados os botões, em qual formato as informações devem ser exportadas e quais informações devem ser exportadas.

Properties / Advanced Properties

Property	Value
Name	exportaVNPLojas
Label	xls
Listeners	[]
Parameters	[]
Component Name	lojasTable
Output Type	xls
HtmlObject	xlsVNPLojas

Figura 24 - Propriedades Componente de Exportação

5.7: Atualização Diária

O banco de dados da Portobello Shop é atualizado diariamente. Sendo assim, as informações que são apresentadas no Dashboard desenvolvido devem também ser atualizadas todos os dias de forma automática.

Para tanto, é necessário executar o processo de ETL, descrito na seção 5.3: deste documento, todos os dias pela manhã. A fim de automatizar esse procedimento, foi desenvolvido um "*job*" responsável por definir a ordem em que as *transformations* desenvolvidas devem ser executadas, como também definir um horário específico para que essa execução seja realizada.

Sendo assim, foi desenvolvido um *job* utilizando a ferramenta PDI como mostra a Figura 25.

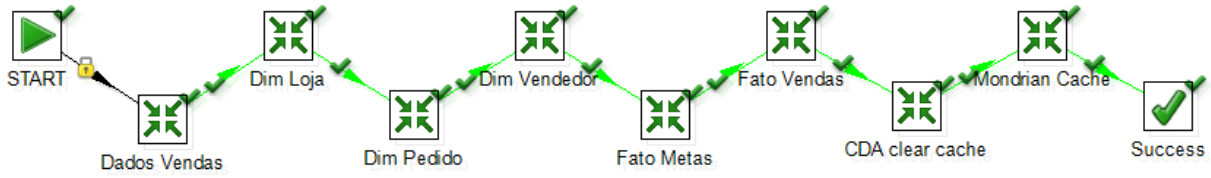

Figura 25 - Job de Atualização Diária

O primeiro *step* é utilizado para definir o horário que será executado este *job*, atualmente definido para todos os dias às 7:45. O último *step* é utilizado para sinalizar o fim da *job*. Já os outros *steps* são responsáveis por executar as

transformações criadas para população das tabelas dimensão e fato, como explicado na seção 5.3: deste documento.

5.8: Usuários e Restrições

Devido à existência de várias lojas da rede Portobello Shop, sendo que algumas pertencem a franqueados diferentes, cada usuário deve ter acesso somente a informações referentes à(s) sua(s) loja(s). Desse modo, foi necessário criar um usuário e senha para cada loja, definindo as restrições de visualização de dados de acordo com o que cada cliente pode ter acesso.

Os usuários e suas restrições são criados separadamente. Primeiro criou-se os *Roles* (ou Cargos), que definem quais informações o usuário terá acesso, como explicado na seção 4.3:. Esses cargos são criados no software *Schema Workbench*, o mesmo utilizado para a criação do cubo de dados. Já os usuários são criados diretamente no servidor de BI, onde é desenvolvido o Dashboard.

5.8.1: Roles

A criação de roles define quais informações do cubo de dados podem ser acessadas para o usuário que possui esse cargo. Entretanto, um usuário pode possuir mais de um *role*. Sendo assim, para esse projeto, foi necessário criar um *role* para cada loja da rede. Dessa forma, caso o usuário necessite de acesso ás informações de mais de uma loja, deve-se apenas adicionar os *roles* referentes a cada loja.

Para criar um cargo, deve-se acessar o *Schema,* criado no software *Schema Workbench*, que contenha o cubo que está sendo utilizado. Adiciona-se então um *role* e criam-se as restrições que o cargo deve ter.

Para facilitar o entendimento da criação de roles, será apresentado a seguir, como exemplo, o código gerado para garantir acesso apenas à Loja A e, logo abaixo, será explicada a funcionalidade do código.

```
<Role name="Loja A">
   <SchemaGrant access="none">
      <CubeGrant cube="PB Vendas Metas" access="all">
```

```
            <HierarchyGrant hierarchy="[Loja.Loja]" topLevel="[Loja.Loja].[Portal]"
access="custom">
                <MemberGrant member="[Loja].[2926].[8]" access="all">
                </MemberGrant>
            </HierarchyGrant>
        </CubeGrant>
    </SchemaGrant>
</Role>
```

Dessa forma, o código mostrado anteriormente como exemplo define um cargo “Loja A” o qual dá acesso apenas a loja [2926].[8], que é a Loja A. Assim como, terá acesso às informações do portal 2926, que é o portal das lojas próprias, porém não terá acesso a todas as lojas da rede, pois o *topLevel* está definido como [Loja.Loja].[Portal].

A Figura 26 mostra o *Schema* resultante após a criação das *Roles*.

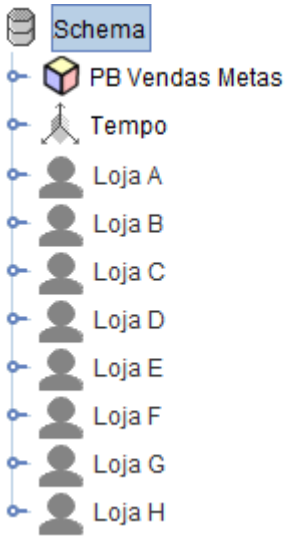

Figura 26 - Schema com Roles Definidos

Capítulo 6: Resultados

Como resultado final do projeto, obteve-se a implantação de um sistema de BI com o intuito de atender usuários de lojas da empresa Portobello Shop, com o objetivo de auxiliar gerentes, franqueados nas tomadas de decisão referentes às vendas de suas lojas.

O projeto inclui a modelagem do *Data Warehouse* utilizado, desenvolvimento do processo de Extração, Transformação e Carga (ETL) das informações, criação de um cubo de dados OLAP e a construção de um Dashboard com informações importantes para as tomadas de decisão.

O início do projeto deu-se por meio da seleção dos indicadores e métricas de desempenho que são mais importantes para realizar o gerenciamento de uma loja. Para isso, as informações necessárias e desejadas para um bom gerenciamento foram discutidas com pessoas ligadas a gestão de lojas.

Após definidos os principais indicadores e métricas de desempenho a serem utilizados, foi modelado o *Data Warehouse* como mostra a Figura 27. Com uma boa modelagem do DW foi possível obter um bom desempenho do sistema, visto que foi utilizada a metodologia de esquema estrela para melhorar o tempo de resposta do sistema.

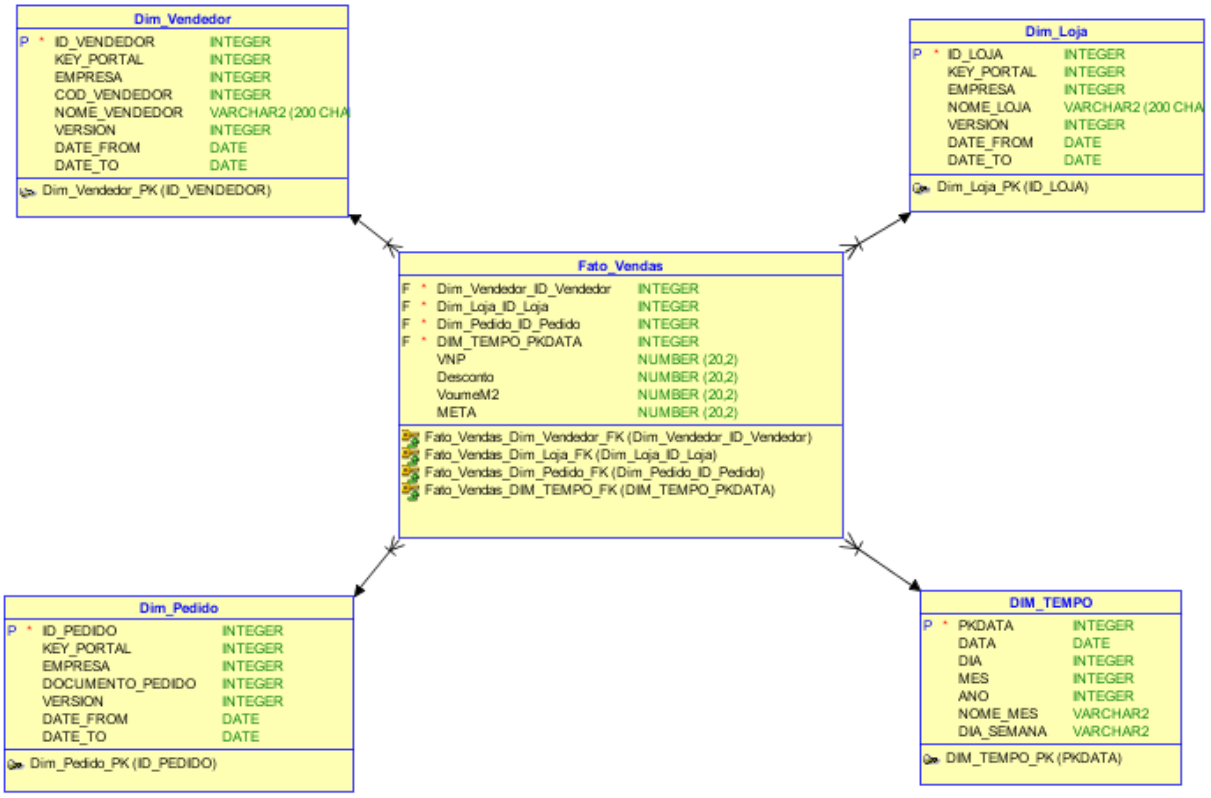

Figura 27 - Resultado da Modelagem do Banco de Dados

Com o DW modelado e criado, foram desenvolvidos todos os processos de ETL necessários para o desenvolvimento do sistema de BI. Dessa maneira, todos os dados necessários foram extraídos do sistema, foram feitas transformações de modo a facilitar o funcionamento do sistema e então foram carregados no DW desenvolvido.

Depois de todos os dados estarem devidamente carregados no DW, foi criado o cubo de dados OLAP. A utilização do cubo OLAP faz com que as análises e visualização dos dados ocorram de forma rápida, consistente e interativa. Além disso, as restrições de acessos às informações a cada usuário foi estabelecida utilizando o cubo de dados.

Desse modo, foi criado o Dashboard apresentado na Figura 28, sendo o resultado final do projeto. Esse Dashboard possibilita a interação do usuário com o painel para que seja possível exibir apenas os dados de interesse do cliente. Além disso, algumas alterações no layout padrão foram realizadas para que o Dashboard apresentasse um design mais amigável e de acordo com as exigências do cliente.

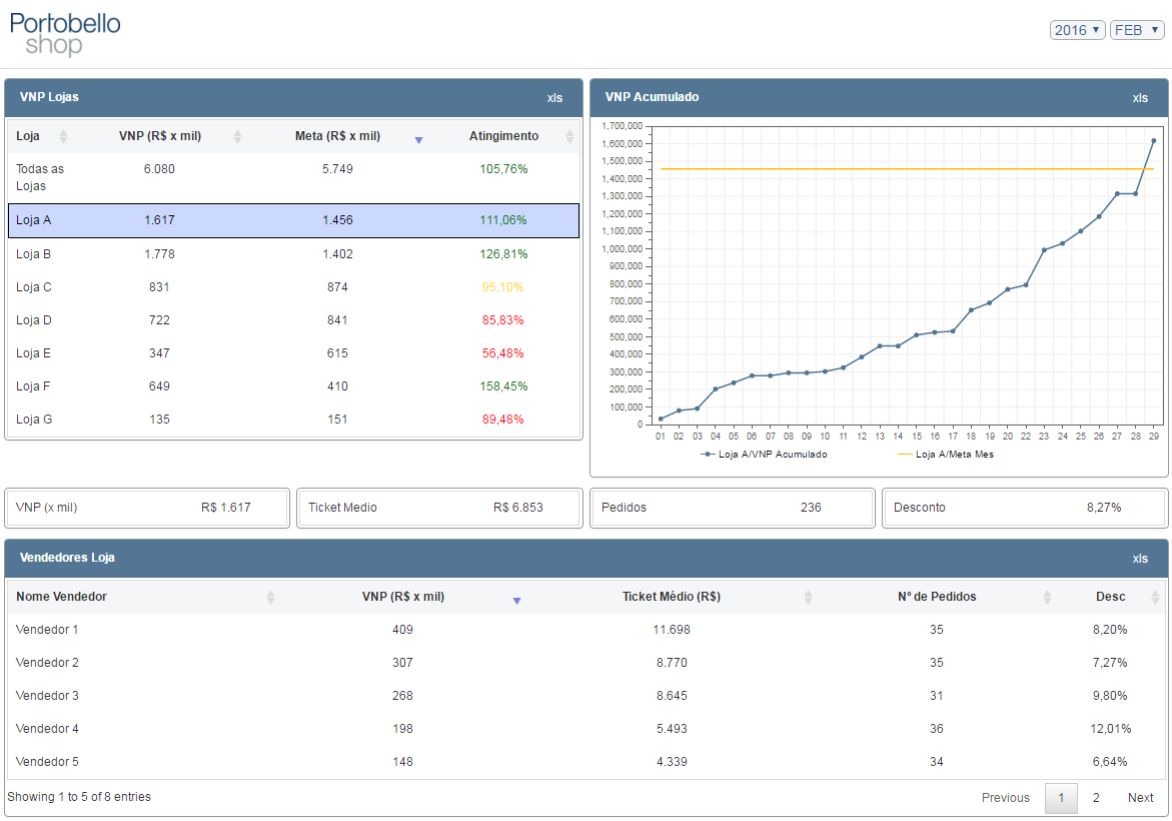

Figura 28 - Dashboard Final

No canto superior esquerdo do painel, como mostra a Figura 28, apresenta-se o logo da empresa cliente. Na mesma linha, no canto superior direito, estão os filtros de data (ano e mês), que são responsáveis por definir o período das informações que devem ser apresentadas no Dashboard.

O primeiro componente à esquerda, VNP Lojas, é a tabela na qual são apresentadas as informações de cada loja, como VNP, meta e atingimento da meta da loja. A última coluna dessa tabela apresenta o percentual de atingimento da meta, as informações apresentadas nessa coluna possuem cores de acordo com o valor a ser mostrado (verde para valores acima de 100%, amarelo para valores entre 90% e 100% e vermelho para valores abaixo de 90%). Por meio dessa tabela, é possível selecionar uma loja clicando com o mouse sobre o nome da loja desejada, filtrando assim as informações apresentadas no painel. A loja selecionada é destacada com um retângulo azul, na Figura 28, por exemplo, está selecionada a loja “Loja A”. Ademais, é possível ordenar as linhas da tabela clicando nas setas

apresentadas nos títulos das colunas, no caso apresentado, a tabela está ordenada de forma decrescente de valores de metas.

O gráfico apresentado no canto superior direito da Figura 28, VNP Acumulado, apresenta o valor em reais no eixo Y (vertical) e os dias do mês selecionado no eixo X (horizontal). As vendas acumuladas da loja selecionada no período definido são representadas através da linha azul. Já a linha laranja do gráfico, apresenta a meta definida para a loja naquele período. No caso apresentado, percebe-se que a loja selecionada atingiu a meta definida, já que as duas linhas do gráfico se cruzam em um ponto.

Os quatro componentes apresentados na linha central do Dashboard, VNP, Ticket Médio, Pedidos e Desconto, são responsáveis por exibir informações importantes, de forma rápida, sobre a loja selecionada. Esses componentes são chamados de *Bullet Points* e também mostram informações de acordo com o período selecionado nos filtros de ano e mês. O componente “VNP (x mil)” apresenta o valor de venda da loja no período especificado, o número apresentado nesse campo deve ser multiplicado por mil para que se obtenha o valor real de venda. O componente “Ticket Médio” exibe o valor médio dos pedidos realizados na loja. O elemento “Pedidos” informa o número de pedidos que foram feitos na loja selecionada. Já o item “Desconto” indica a porcentagem de desconto concedida.

O último componente do Dashboard, a tabela “Vendedores Loja” é responsável por apresentar informações específicas de cada vendedor da loja seleciona no período definido. Essa tabela exibe o valor de venda realizado (VNP), a média de valor dos pedidos (Ticket Médio), o número de pedidos efetuados (Nº de Pedidos) e a porcentagem de desconto concedida (Desc) por cada vendedor. Assim como a tabela de loja, essa pode ser ordenada de acordo com os valores apresentados em suas colunas, bastando clicar nas setas contidas ao lado do título de cada coluna.

Além disso, é possível exportar as informações contidas em cada tabela ou gráfico clicando nos botões “xls” apresentados nos cantos superiores de cada componente. Ao clicar em um desses botões, é iniciado um download automático de uma tabela em formato “xls” contendo as informações do componente específico.

O Dashboard apresentado na Figura 28 é exibido quando o usuário acessa o sistema através de um computador. Porém, caso o cliente tenha interesse em verificar as informações referentes à sua loja através de um aparelho mobile (celular ou tablet), o Dashboard é capaz de adaptar-se para que seja exibido de uma forma adequada, visto que estes aparelhos possuem telas menores, o que dificulta a visualização das informações. Sendo assim, quando acessado por um celular, por exemplo, o Dashboard é apresentado como mostrado na Figura 29.

Figura 29 - Layout Dashboard Mobile

Com o Dashboard criado, foram realizados alguns testes para verificação do funcionamento do sistema. Foi liberado para alguns usuários o acesso ao servidor para que eles realizarem alguns testes e também fisessem sugestões de melhorias.

Atualmente, não foi encontrado nenhum problema e as sugestões que foram dadas em relação ao layout já estão aplicadas na como mostram a Figura 28 e Figura 29.

Sendo o objetivo geral do trabalho, auxiliar a gerência das lojas Portobello Shop nas tomadas de decisão, o sistema de BI desenvolvido oferece a visualização e velocidade de obtenção de resultados de análise de forma mais eficaz. Visto que, atualmente, o gerenciamento das lojas se dá através de cruzamento de tabelas, o que despende muito mais tempo.

Sendo assim, a utilização do sistema de BI desenvolvido auxiliará os usuários em suas tomadas de decisão. Além disso, o sistema de BI utiliza dados exatos e confiáveis, evitando problemas de análises de relatórios sobrepostos com dados duvidosos. Dessa forma, diminui-se ainda mais o tempo de análise de dados, já que não será gasto tempo para descobrir quais dados são mais confiáveis.

Consequentemente, a diminuição do tempo de análise de dados torna a organização mais preparada para concorrer no mercado. Tendo em vista que, com a diminuição desse tempo, é possível tomar decisões de forma mais rápida e achar soluções antes mesmo de ocasionar um problema relevante.

Capítulo 7: Conclusões e Perspectivas

O desenvolvimento desse projeto ocorreu na empresa FernBI e atingiu o resultado esperado pela empresa, o qual consistia em avaliar a viabilidade e escalabilidade da implantação de um sistema de BI para clientes com um grande número de usuários.

O sistema de BI desenvolvido com a finalidade de auxiliar os gerentes e franqueados da rede Portobello Shop está em fase de aprovação e ainda não foi recebida uma resposta por parte da empresa.

Durante o desenvolvimento desse sistema de BI utilizando a ferramenta Pentaho, várias dificuldades foram encontradas, visto que não é um assunto muito abordado no curso de Engenharia de Controle e Automação.

Por ser o primeiro projeto desenvolvido em Pentaho na empresa FernBI, demandou um esforço ainda maior, uma vez que nenhum outro funcionário da empresa é familiarizado com a ferramenta, o que dificultava a solução de problemas relacionados especificamente ao uso da ferramenta.

O sistema de BI desenvolvido neste projeto inclui, até o momento, sete lojas da rede e possui uma grande capacidade de auxiliar o cliente no gerenciamento dessas lojas, exibindo informações pertinentes para tomadas de decisão de forma rápida e confiável.

Além de atender as sete lojas que já utilizaram o sistema de Business Intelligence, o projeto desenvolvido tem potencial para atender as mais de 140 lojas presentes na rede Portobello Shop.

O presente projeto inclui a apresentação de um sistema de Business Intelligence com as informações mais importantes para o cliente. Sendo base para o desenvolvimento de novos módulos a serem implementados. Além disso, um novo painel com outras informações já está sendo desenvolvido a fim de facilitar ainda mais o gerenciamento das lojas com o objetivo de, cada vez mais, melhorar o desempenho geral da rede Portobello Shop.

O presente projeto foi aprovado e comprado pela Portobello Shop. Dessa forma, o sistema desenvolvido será expandido para atender as mais de 140 lojas da rede.

Bibliografia:

[1] M. A. O. Neves, "Tudo sobre indicadores de desempenho em logística" Revista Mundo Logística, 2009.

[2] M. S. Almeida, M Ishikawa, J. Reinschmidt e T. Roeber, "Getting Started with DataWarehouse and Business Intelligence", Agosto 1999.

[3] R. Kimball e M. Ross, "The Data Warehouse Toolkit Third Edition: The Definitive Guide to Dimensional Modeling", 2013.

[4] H. J. WATSON, "Building Executive Information Systems and Other Decision Support Applications", 1997.

[5] E. Thomsen, "Olap Solutions Second Edition: Building Multidimensional Information Systems", 2002.

[6] "Mondrian 2.2.2 Technical Guide Developing OLAP solutions with Mondrian", Março 2007.

[7] R. Percival, "The Complete Web Developer Course" Curso em http://www.udemy.com

[8] Manual QlikView

[9] http://www.qlik.com/pt-br

[10] http://www.pentaho.com/

Apêndice:

vendasLojaQuery

```
WITH
MEMBER [Measures].[Meta Mes] AS 'Sum(PeriodsToDate([Tempo].[Mes]),
[Measures].[Meta])'
member [Measures].[% Meta] AS '(([Measures].[VNP])/[Measures].[Meta
Mes])*100'
MEMBER [Measures].[VNP abrev] AS '[Measures].[VNP]/1000'
MEMBER [Measures].[Meta abrev] AS '[Measures].[Meta Mes]/1000'
SET [~FILTER] AS
   {${mesParameter}}
SET [~ROWS] AS
   {[Loja].[Todas as Lojas], [Loja].[Nome Loja].Members}
SELECT
   NON EMPTY {[Measures].[VNP abrev],[Measures].[Meta abrev],
[Measures].[% Meta]} ON COLUMNS,
   NON EMPTY [~ROWS] ON ROWS
FROM [PB Vendas Metas]
WHERE [~FILTER]
```

vnpLojaAcumQuery

```
WITH
MEMBER [Measures].[VNP Acumulado] AS
'Sum(PeriodsToDate([Tempo].[Mes]), [Measures].[VNP])'
MEMBER [Measures].[Meta Mes] AS 'Sum(PeriodsToDate([Tempo].[Mes]),
[Measures].[Meta])'
SET [~COLUMNS] AS
   {${loja}}
SET [~ROWS] AS
   {${mesParameter}.children}
SELECT
   CrossJoin([~COLUMNS], {[Measures].[VNP Acumulado], [Measures].[Meta
Mes]}) ON COLUMNS,
   [~ROWS] ON ROWS
FROM [PB Vendas Metas]
```

vendedoresTableQuery

```
WITH
MEMBER [Measures].[VNP abrev] AS '[Measures].[VNP]/1000'
MEMBER [Measures].[TM abrev] AS '[Measures].[Ticket Medio]/1000'
MEMBER [Measures].[Porc Desconto] AS '[Measures].[Porcentagem
Desconto]*100'
```

```
SET [~ROWS] AS
   {[Vendedor].[Nome Vendedor].Members}
SELECT
NON EMPTY {[Measures].[VNP abrev], [Measures].[Ticket Medio],
[Measures].[Numero de Pedidos], [Measures].[Porc Desconto]} ON COLUMNS,
NON EMPTY [~ROWS] ON ROWS
FROM [PB Vendas Metas]
WHERE CrossJoin({${mesParameter}}, {${loja}})
```

vnpLojaQuery

```
WITH
MEMBER [Measures].[VNP (x mil)] AS '[Measures].[VNP]/1000'
SET [~ROWS] AS
   {${loja}}
SELECT
NON EMPTY {[Measures].[VNP (x mil)]} ON ROWS,
NON EMPTY [~ROWS] ON COLUMNS
FROM [PB Vendas Metas]
WHERE ({${mesParameter}})
```

ticketMedioQuery

```
WITH
MEMBER [Measures].[Ticket Médio] AS '[Measures].[Ticket Medio]'
SET [~ROWS] AS
   {${loja}}
SELECT
NON EMPTY {[Measures].[Ticket Medio]} ON ROWS,
NON EMPTY [~ROWS] ON COLUMNS
FROM [PB Vendas Metas]
WHERE (${mesParameter})
```

noPedidosQuery

```
WITH
MEMBER [Measures].[Pedidos] AS '[Measures].[Numero de Pedidos]'
SET [~ROWS] AS
   {${loja}}
SELECT
NON EMPTY {[Measures].[Pedidos]} ON ROWS,
NON EMPTY [~ROWS] ON COLUMNS
FROM [PB Vendas Metas]
WHERE (${mesParameter})
```

porcDescontoQuery

```
WITH
```

```
MEMBER [Measures].[Desconto] AS '[Measures].[Porcentagem Desconto] *
100'
SET [~ROWS] AS
   {${loja}}
SELECT
NON EMPTY {[Measures].[Desconto]} ON ROWS,
NON EMPTY [~ROWS] ON COLUMNS
FROM [PB Vendas Metas]
WHERE (${mesParameter})
```

anoSelectorQuery

```
with member [Measures].[Name] as '[Tempo].CurrentMember.UniqueName'

select TopCount( filter({Descendants([Tempo].[Todos os Tempos]
,[Tempo].[Ano])}, not isempty(([Tempo].CurrentMember)) ) , 50) on ROWS,
 {[Measures].[Name]} on Columns
FROM [PB Vendas Metas]
 WHERE ({${loja}})
```

mesSelectorQuery

```
with
 member [Measures].[Name] as '[Tempo].CurrentMember.UniqueName'

select TopCount( filter({Descendants(${anoParameter} ,[Tempo].[Mes])}, not
isempty(([Tempo].CurrentMember)) ) , 50) on ROWS,
 {[Measures].[Name]} on Columns
FROM [PB Vendas Metas]
 WHERE ({${loja}})
```

Código JS para adaptação Bootstrap:

```
function f(){
var myself = this;
 // Define largura inicial
 myself.chartDefinition.width = myself.placeholder().width();

 // Redefine o tamanho somente na primeira execução do componente
 if (!this.resizeHandlerAttached){

   // Certifica que a renderização só é acionada após parar de mudar o
tamanho
   var debouncedResize = _.debounce(function(){

     // Mostra o gráfico novamente
     myself.placeholder().children().css('visibility','visible');
```

```
            // Muda o tamanho do gráfico
            myself.chartDefinition.width = myself.placeholder().width();
            myself.render( myself.query.lastResults() );
          }, 200);

          // vincula o manipulador de tamanho
          $(window).resize(function(){

            // só aciona o redimensionamento se a janela mudar de tamanho
            if ( myself.chartDefinition.width != myself.placeholder().width()){

              // esconde temporariamente o gráfico para não ter overflow
              myself.placeholder().children().css('visibility','hidden');

              // aciona o redimensionamento
              debouncedResize();
            }
          });

          this.resizeHandlerAttached = true;
        }

      }
```

Código CSS desenvolvido:

```
      /*****************************Formatacao Dashboard*******************************/

      .col-xs-1, .col-sm-1, .col-md-1, .col-lg-1, .col-xs-2, .col-sm-2, .col-md-2, .col-lg-2, .col-xs-3, .col-sm-3, .col-md-3, .col-lg-3, .col-xs-4, .col-sm-4, .col-md-4, .col-lg-4, .col-xs-5, .col-sm-5, .col-md-5, .col-lg-5, .col-xs-6, .col-sm-6, .col-md-6, .col-lg-6, .col-xs-7, .col-sm-7, .col-md-7, .col-lg-7, .col-xs-8, .col-sm-8, .col-md-8, .col-lg-8, .col-xs-9, .col-sm-9, .col-md-9, .col-lg-9, .col-xs-10, .col-sm-10, .col-md-10, .col-lg-10, .col-xs-11, .col-sm-11, .col-md-11, .col-lg-11, .col-xs-12, .col-sm-12, .col-md-12, .col-lg-12
{
          position: relative;
          min-height: 1px;
          padding-right: 3px;
          padding-left: 3px;
          margin-bottom: -10px;
      }

      body {
          font-size: 12px;
      }

      .panel-body {
          padding: 2px;
      }
```

```
/*****************************Barra de Navegacao*****************************/

img {
    padding: 10px;
    padding-top: 15px;
    vertical-align: middle;
}

.navbar-inverse {
    background-color: #FFFFFF;
    border-color: rgba(205, 207, 208, 0.68);
    max-width: 1200px;
    margin: 0 auto;
}

.navbar-nav.navbar-right:last-child {
    padding: 20px;
    margin-right: -15px;
}

.botao {
    padding-right: 2px;
}

.navbar-inverse .navbar-collapse, .navbar-inverse .navbar-form {
    border-color: rgba(205, 207, 208, 0.68);
}

.navbar-inverse .navbar-toggle .icon-bar {
    background-color: #527694;
}

.navbar-inverse .navbar-toggle:hover, .navbar-inverse .navbar-toggle:focus {
    background-color: #FFFFFF;
}

.navbar-inverse .navbar-toggle {
    border-color: #315998;
}

input, button, select, textarea {
    font-family: inherit;
    font-size: 14px;
    line-height: inherit;
    border-radius: 5px;
    color: #315998;
    background-color: #F5F6F7;
}
```

```
/************************Formatacao Tabelas*********************************/

.exportButton{
    float: right;
    background-color: transparent;
    margin-top: -18px;
}

.table-bordered > thead > tr > th, .table-bordered > tbody > tr > th, .table-
bordered > tfoot > tr > th, .table-bordered > thead > tr > td,.table-bordered > tbody >
tr > td, .table-bordered > tfoot > tr > td {
    border: 0px solid #ddd;
}

.visitsTable table{
    cursor: pointer;
}

.panel-primary > .panel-heading {
    color: #fff;
    background-color: #527694;
    border-color: #527694;
}

.panel-primary {
    border-color: #959ca1;
}
.tabela thead {
    display: none;
    text-align: right;
}

.tabela .panel-primary > .panel-heading {
    display: none;
}

.panel-primary > .panel-footer {
    display: none;
}

.r1 .panel-body {
    padding: 2px;
}

.tabela .panel-body {
    padding: 2px;
}

.tabela table td.column1 {
```

```
        width: 120px;
    }

    tr.selected > td{
        border: 1px solid #656666;
    }

    .dataTable tr{
        background-color: #F5F6F7 !important;
    }

    .visitsTable table tr td{
        background-color: #FFFFFF !important;
    }

    .numeric, .numericBR, .numericBRcif, .numericBRporc, .numericBRporcAting,
.value, .numericBRsd{
        text-align: center;
    }

    /*****************************Select Table*********************************/

    .visitsTable table{

    }

    .visitsTable table tr:hover > td.column0{
        padding-left: 7px !important;
        border-left: 1px solid #656666 !important;
        border-top: 1px solid #656666 !important;
        border-bottom: 1px solid #656666 !important;
    }

    .visitsTable table tr:hover > td.column1{

        border-top: 1px solid #656666 !important;
        border-bottom: 1px solid #656666 !important;
        border-left: 1px transparent !important;
    }

    .visitsTable table tr:hover > td.column2{

        border-top: 1px solid #656666 !important;
        border-bottom: 1px solid #656666 !important;
        border-left: 1px transparent !important;
        border-right: 1px transparent !important;
    }

    .visitsTable table tr:hover > td.column3{
```

```
    padding-right: 7px !important;
    border-right: 1px solid #656666 !important;
    border-top: 1px solid #656666 !important;
    border-bottom: 1px solid #656666 !important;
}


.visitsTable table tr.selected > td.column0{
    padding-left: 7px !important;
    border-left: 1px solid  !important;
    border-top: 1px solid  !important;
    border-bottom: 1px solid  !important;
}

.visitsTable table tr.selected > td.column1{

    border-top: 1px solid  !important;
    border-bottom: 1px solid  !important;
    border-left: 1px transparent !important;
}

.visitsTable table tr.selected > td.column2{

    border-top: 1px solid  !important;
    border-bottom: 1px solid  !important;
    border-left: 1px transparent !important;
}

.visitsTable table tr.selected > td.column3{
    padding-right: 7px !important;
    border-right: 1px solid  !important;
    border-top: 1px solid  !important;
    border-bottom: 1px solid  !important;
}

.visitsTable table tr.selected > td{
    background-color: #ccd9ff !important;
    color: black;
}

.visitsTable table tr{
    padding: 0px;
}


.table-striped>tbody>tr:nth-child(odd)>td,
.table-striped>tbody>tr:nth-child(odd)>th,
.table-striped>tbody>tr:nth-child(even)>td,
```

```
.table-striped>tbody>tr:nth-child(even)>th{
    background-color: #FFFFFF;
}
```

Código JS para formatação numérica:

```
/**  inicio addIn numericBR **/
var numericBR = {
name: "numericBR",
label: "Numeric BR",
defaults: {
decimal: true,
textFormat: function(v, st, opt) {
var nStr= v.toFixed(2)+'';
x = nStr.split('.');
x1 = x[0];
x2 = x.length > 1 ? ',' + x[1] : '';
var rgx = /(\d+)(\d{3})/;
while (rgx.test(x1)) {
x1 = x1.replace(rgx, '$1' + '.' + '$2');
}
return x1 + (opt.decimal ? x2 : '');
}
},
init: function(){
$.fn.dataTableExt.oSort[this.name+'-asc'] = $.fn.dataTableExt.oSort['numeric-
asc'];
$.fn.dataTableExt.oSort[this.name+'-desc'] =
$.fn.dataTableExt.oSort['numeric-desc'];
},

implementation: function(tgt, st, opt){
var text = opt.textFormat.call(this, st.value, st, opt);
$(tgt).empty().append("<span style='text-align:right;'>"+text+"</span>");
}

};
Dashboards.registerAddIn("Table", "colType", new AddIn(numericBR));
/** fim addIn numericBR **/




/** inicio addIn numericBRsd **/
var numericBRsd = {
name: "numericBRsd",
label: "Numeric BR Sem Decimais",
defaults: {
```

```
decimal: true,
textFormat: function(v, st, opt) {
var nStr= v.toFixed(0)+'';
x = nStr.split('.');
x1 = x[0];
x2 = x.length > 1 ? ',' + x[1] : '';
var rgx = /(\d+)(\d{3})/;
while (rgx.test(x1)) {
x1 = x1.replace(rgx, '$1' + '.' + '$2');
}
return x1;
}
},
init: function(){
$.fn.dataTableExt.oSort[this.name+'-asc'] = $.fn.dataTableExt.oSort['numeric-
asc'];
$.fn.dataTableExt.oSort[this.name+'-desc'] =
$.fn.dataTableExt.oSort['numeric-desc'];
},

implementation: function(tgt, st, opt){
var text = opt.textFormat.call(this, st.value, st, opt);
$(tgt).empty().append("<span style='text-align:right;'>"+text+"</span>");
}

};
Dashboards.registerAddIn("Table", "colType", new AddIn(numericBRsd));
/** fim addIn numericBRsd **/



/** inicio addIn numericBRporc **/
var numericBRporc = {
name: "numericBRporc",
label: "Numeric BR Porcentagem",
defaults: {
decimal: true,
textFormat: function(v, st, opt) {
var nStr= v.toFixed(2)+'';
x = nStr.split('.');
x1 = x[0];
x2 = x.length > 1 ? ',' + x[1] : '';
var rgx = /(\d+)(\d{3})/;
while (rgx.test(x1)) {
x1 = x1.replace(rgx, '$1' + '.' + '$2');
}
return x1 + (opt.decimal ? x2 : '') + '%';
}
},
```

```
	init: function(){
	$.fn.dataTableExt.oSort[this.name+'-asc'] = $.fn.dataTableExt.oSort['numeric-
asc'];
	$.fn.dataTableExt.oSort[this.name+'-desc'] =
$.fn.dataTableExt.oSort['numeric-desc'];
	},

	implementation: function(tgt, st, opt){
	var text = opt.textFormat.call(this, st.value, st, opt);
	$(tgt).empty().append("<span style='text-align:right;'>"+text+"</span>");
	}

	};
	Dashboards.registerAddIn("Table", "colType", new AddIn(numericBRporc));
	/** fim addIn numericBRporc **/


	/** inicio addIn numericBRcif **/
	var numericBRcif = {
	name: "numericBRcif",
	label: "Numeric BR R$",
	defaults: {
	decimal: true,
	textFormat: function(v, st, opt) {
	var nStr= v.toFixed(0)+'';
	x = nStr.split('.');
	x1 = x[0];
	x2 = x.length > 1 ? ',' + x[1] : '';
	var rgx = /(\d+)(\d{3})/;
	while (rgx.test(x1)) {
	x1 = x1.replace(rgx, '$1' + '.' + '$2');
	}
	return 'R$ ' + x1;
	}
	},
	init: function(){
	$.fn.dataTableExt.oSort[this.name+'-asc'] = $.fn.dataTableExt.oSort['numeric-
asc'];
	$.fn.dataTableExt.oSort[this.name+'-desc'] =
$.fn.dataTableExt.oSort['numeric-desc'];
	},

	implementation: function(tgt, st, opt){
	var text = opt.textFormat.call(this, st.value, st, opt);
	$(tgt).empty().append("<span style='text-align:right;'>"+text+"</span>");
	}

	};
	Dashboards.registerAddIn("Table", "colType", new AddIn(numericBRcif));
```

```
/** fim addIn numericBR$ **/


/** inicio addIn numericBRporcAting **/
var numericBRporcAting = {
name: "numericBRporcAting",
label: "Numeric BR Porcentagem Atingimento",
defaults: {
decimal: true,
textFormat: function(v, st, opt) {
var nStr= v.toFixed(2)+'';
x = nStr.split('.');
x1 = x[0];
x2 = x.length > 1 ? ',' + x[1] : '';
var rgx = /(\d+)(\d{3})/;
while (rgx.test(x1)) {
x1 = x1.replace(rgx, '$1' + '.' + '$2');
}
y = x1 + (opt.decimal ? x2 : '') + '%';
if(v<90){
   return y.fontcolor("red");
}else{
   if(v<100){
      return y.fontcolor("#FFDD22");
   }else{
      return y.fontcolor("green");
   }
}

}
},
init: function(){
$.fn.dataTableExt.oSort[this.name+'-asc'] = $.fn.dataTableExt.oSort['numeric-
asc'];
$.fn.dataTableExt.oSort[this.name+'-desc'] =
$.fn.dataTableExt.oSort['numeric-desc'];
},

implementation: function(tgt, st, opt){
var text = opt.textFormat.call(this, st.value, st, opt);
$(tgt).empty().append("<span style='text-align:right;'>"+text+"</span>");
}

};
Dashboards.registerAddIn("Table", "colType", new
AddIn(numericBRporcAting));
/** fim addIn numericBRporcAting **/
```

Código JS desenvolvido para interação com tabela de lojas:

```
function f(){

    $('#visitsTableTable tbody tr:first').addClass("selected");

   var ph = $("#" + this.htmlObject + " table tbody tr");
   ph.bind("click",function(){

      var row = $(this);
      var v = row.find("td:eq(0)").text();
      if(v == "Todas as Lojas"){
         var loja = '[Loja].['+v+']';
      }else{
         var loja = '[Loja].[Nome Loja].['+v+']';
      }

      Dashboards.fireChange("lojaParameter",loja);

      // Remove a marca de seleção das outras linhas e marca essa
      row.parent().find("tr").removeClass("selected");
      row.addClass("selected");

   });

   // Remove classe selecionada de toads as linhas e adiciona na clicada
   ph.removeClass('selected');

   ph.each( function(idx, elem){
      var lojaTabela = $(elem).find('.title').text();
      if(lojaTabela == "Todas as Lojas"){
         var lojaSelecionada = '[Loja].['+lojaTabela+']';
      }else{
         var lojaSelecionada = '[Loja].[Nome Loja].['+lojaTabela+']';
      }
      if ( lojaSelecionada == lojaParameter ){
         $(elem).addClass('selected');
      }
   } );

}
```

Código JS para iniciação do Dashboard com o mês atual:

```
function(){
   var monthNames = ["JAN", "FEV", "MAR", "ABR", "MAI", "JUN", "JUL",
"AGO", "SET", "OUT", "NOV", "DEZ"];
   var d = new Date();
   var mes = '[Tempo].['+(d.getFullYear())+'].['+monthNames[d.getMonth()]+']';
   return mes;
 }
```