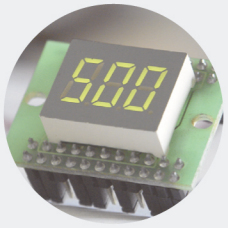
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






DESENVOLVIMENTO DE PROCESSOS DE BUSINESS INTELLIGENCE EM FERRAMENTAS DE BIG DATA

TIAGO DA SILVA VIEIRA
novembro de 2018

POLITÉCNICO
DO PORTO

DESENVOLVIMENTO DE PROCESSOS DE *BUSINESS INTELLIGENCE* EM FERRAMENTAS DE *BIG DATA*

Tiago da Silva Vieira
Outubro de 2018

Departamento de Engenharia Eletrotécnica
Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização em Sistemas e Planeamento Industrial

Relatório elaborado para satisfação parcial dos requisitos da Unidade Curricular de Tese/Dissertação do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Tiago da Silva Vieira, Nº 1160017, 1160017@isep.ipp.pt
Orientação científica: Profª Cecília Maria do Rio Fernandes Moreira Reis, cmr@isep.ipp.pt
Empresa: Sonae MC – BIT (*Business Information Tecnology*)
Supervisão: Frederico da Silva Couceiro.

Departamento de Engenharia Eletrotécnica
Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização em Sistemas e Planeamento Industrial

2018

Agradecimentos

Aproveito esta oportunidade para expressar a minha gratidão a todos aqueles que me ajudaram a concluir mais esta etapa da minha vida.

À minha orientadora, pela partilha de conhecimentos e ajuda prestada, que estou certo que em muito contribuíram para o término com sucesso desta dissertação.

Quero agradecer aos meus colegas de mestrado, pela partilha de experiências que vivemos e pelas amizades que construí para a vida.

Aos meus amigos, que souberam aguardar a minha ausência e que sempre estiveram na retaguarda a torcer pelos meus sucessos.

Ao meu irmão, pelo ânimo e incentivo que me transmitiu, por toda a confiança que sempre depositou em mim.

Aos meus pais, pela paciência que tiveram ao longo destes últimos anos de estudos e, principalmente, destes últimos dois anos, que tantos sacrifícios exigiram.

Não posso deixar de endereçar um enorme agradecimento aos meus avós, pela dedicação de uma vida, pelos esforços que fizeram na minha educação e formação académica.

Aproveito para agradecer aos meus "sogros" e ao Zé Pedro, por serem a segunda família que escolhi e por me fazerem sentir sempre em casa.

Por último, quero agradecer à Beatriz, por todo o acompanhamento pessoal e profissional, por sempre me incentivar a desafiar-me e a chegar mais longe. Que esta meta seja mais uma no caminho que juntos construímos.

Resumo

Na Economia atual, obter informação com qualidade e atempadamente é uma mais valia nas organizações para que estas alcancem uma vantagem competitiva, principalmente quando se trata de grandes empresas com portefólios de negócios alargados.

O propósito desta dissertação consiste na implementação de processos de BI em ferramentas de Big Data, de forma a comprovar a sua mais-valia no auxílio à tomada de decisão.

No âmbito da implementação do projeto Profit & Loss Comercial, na Sonae MC, mais especificamente BIT, foi identificada a necessidade de alterar o atual processo de disponibilização de receitas comercias, mais especificamente as relativas com processos marginais e promocionais.

O pedido advém da necessidade de alterar o comportamento do processo atual, de forma a permitir uma visão analítica da informação financeira atualmente existente. É então intuito distribuir os valores de receitas cadastrados pelos utilizadores, de forma a que estes espelhem o seu impacto na rentabilidade comercial diária de toda a organização.

Para tal, foram definidos algoritmos de distribuição juntamente com a equipa de negócio Sonae, que visam ratear as receitas cadastradas de acordo com a disponibilidade do comercial, de uma forma equilibrada pelos dias de afetação da receita e peso diário de cada loja na empresa como um todo.

Os resultados obtidos possuem um grau de confiança e sucesso grande pois os processos implementados cumprem todos os requisitos e com um desempenho excelente. Logicamente que para atingir tempos aceitáveis foi necessário afinar os processos através de algumas técnicas, havendo ainda assim a necessidade de realizar uma verificação e melhoria constante.

Palavras-Chave: *Business Intelligence, Data Warehouse, Big Data, Hadoop, MapReduce, Spark, Hive, Scala.*

Abstract

In the current economy, having information with quality and timely is a profit in organizations so they can achieve a competitive advantage, especially when it comes to large companies with extended portfolios.

The purpose of this thesis is to implement BI processes in Big Data tools to prove its added value in aiding decision making. As part of Commercial Profit & Loss implementation in Sonae MC, more specifically BIT, we identified the need to change the current process of making commercial revenues, namely those related to marginal and promotional processes.

The first step of this implementation was the characterization and specification of the problem, thus realizing the real needs and requirements of the commercial teams. Subsequently, the processes were implemented, executed and validated in the existing Big Data tools on the organization. At last, some conclusions were announced such as questions for further investigation.

The request comes from the need to change the behavior of the current process, to add an analytical view of the financial information that currently exists. It is then intended to distribute the revenue values registered by users, so they reflect their impact on the daily business profitability of the entire organization.

To this end, distribution algorithms were defined together with the Sonae business team, which aim to split the revenues registered according to the availability of the commercial, in a manner balanced by the days of revenue allocation and daily weight of each store in the company.

The results obtained have a degree of confidence and great success because the processes implemented meet all the requirements and with an excellent performance. Of course, to achieve acceptable times it was necessary to fine-tune the processes through some techniques, but there is still a need for constant verification and improvement.

Keywords: *Business Intelligence, Data Warehouse, Big Data, Hadoop, MapReduce, Spark, Hive, Scala.*

Índice

Índice de Ilustrações

Índice de Tabelas

Acrónimos

BI	*Business Intelligence*
SQL	Structured *Query* Language
NoSQL	Not Only SQL
HQL	Hive *Query* Language
ERP	Enterprise Resource Planning
PC	Personal Computer
XML	eXtensible Markup Language
IBM	International Business Machine
HDFS	*Hadoop* Distributed File System
RDBMs	Relational Database Management System
PK	Primary Key
NK	Natural key
SK	Surrogate key
SCD	Slowly Changing Dimension
ETL	Extraction, Transformation, Loading
ELT	Extraction, Loading, Transformation
API	Application Programming Interface
PLA	Profit and Loss Analysis
IDE	Integrated Development Environment
BIT	Business Information Technology
EDW	Enterprise Data Warehouse

1. INTRODUÇÃO

1.1. CONTEXTO DE INVESTIGAÇÃO: A SONAE S.G.P.S., S.A.

A Sonae SGPS, S.A. é uma empresa multinacional portuguesa, sediada na Maia, e representa o maior empregador português com cerca de 46 mil colaboradores, distribuídos por um diversificado portefólio de negócios.

A Sonae foi criada em 1959 pelo empresário, banqueiro e mecenas Afonso Pinto de Magalhães. A entrada de Belmiro de Azevedo na Sonae remonta ao ano de 1965 e é em 1982 que Afonso Pinto de Magalhães lhe cede 16% da empresa. Posteriormente, após o falecimento do fundador, Belmiro de Azevedo atinge a maioria do capital com 54,6% e assume o controlo da Sonae (Sonae 2018).

Durante os anos 80, a Sonae iniciou o seu crescimento. Em 1985 foi criada a Sonae Investimentos S.G.P.S., S.A., bem como a abertura do seu primeiro hipermercado em Portugal, o Continente de Matosinhos. A estratégia de diversificação de negócio do grupo coincide também com esta data, através de aquisições e criação de novos investimentos, que viriam a culminar na entrada da Sonae na bolsa de valores de Lisboa, em 2000.

Em março de 2015, Belmiro de Azevedo anunciou a saída de chairman da Sonae e passa a sua posição ao filho Paulo de Azevedo, que assumia a liderança do grupo desde 2007.

Em 2018, Cláudia Azevedo – filha de Belmiro de Azevedo – é eleita presidente executiva da Sonae e irá iniciar funções no início de 2019.

A figura 1 evidencia alguns indicadores-chave de desempenho do grupo que dizem respeito aos resultados anuais de 2017.

Figura 1 - Alguns KPI's financeiros da Sonae (dados atualizados a março de 2018) (Sonae 2018).

Com a missão de "criar valor económico e social a longo prazo, levando os benefícios do progresso e da inovação a um número crescente de pessoas", a Sonae está em todos continentes, com presença num total de 91 países (Sonae 2018).

O seu portefólio de negócios é bastante alargado, atuando em diversos setores, entre os quais se destacam o retalho, serviços financeiros, gestão de centros comerciais, software e sistemas de informação, media e telecomunicações.

A figura 2 destaca as principais unidades de negócio da Sonae S.G.P.S., S.A.:

Figura 2 - Portefólio de negócios da Sonae S.G.P.S., S.A. (Sonae 2018)

No que à Sonae MC diz respeito, trata-se de uma empresa na área do retalho alimentar, saúde e bem-estar. A Sonae MC é líder do mercado nacional com um conjunto de formatos de retalho distintos: Continente (hipermercados), Continente Modelo e Continente Bom dia (supermercados de conveniência), Meu Super (lojas de proximidade em formato franchising), Bom Bocado, Bagga (cafetarias e restaurantes), Go Natural (supermercados e restaurantes saudáveis), Make Notes, Note! (livraria/papelaria), ZU (produtos e serviços para cães e gatos), Well's (saúde, bem-estar e ótica) e Dr. Well's (clínicas medicina dentária e medicina estética) (Sonae 2018).

Por sua vez, a BIT – *Business Information Technology* consiste na área de Sistemas de Informação da Sonae, com o intuito de criar ferramentas de gestão e decisão rápidas e coesas para todo o grupo Sonae.

A BIT tem procurado desenvolver soluções para as diferentes referências da cadeia de valor, das lojas aos entrepostos, ao criar novas soluções e proporcionar uma experiência mais inclusiva nas diferentes insígnias da Sonae, para procurar responder aos desafios diários dos consumidores. Além disso, trabalha diferentes tecnologias num ambiente criativo e dinâmico, com o intuito de que este clima de colaboração e parceria fomente a troca de conhecimentos e possibilite a utilização dos sistemas de informação como uma alavanca para a vantagem competitiva das diferentes insígnias do grupo Sonae (Gaboleiro 2016).

1.2. CONTEXTUALIZAÇÃO

Atualmente, graças sobretudo à globalização dos mercados e aos avanços tecnológicos, é imperativo para as organizações obter informação com qualidade e atempadamente, de modo a que estas possam alcançar uma vantagem competitiva, principalmente quando se trata de grandes empresas com portefólios de negócios diversificados.

Neste sentido, face à volatilidade da economia, o *Business Intelligence* tornou-se uma ferramenta poderosa nas organizações. Este veio permitir às empresas a tomada de decisão de uma forma mais conscienciosa e cautelosa, contribuindo para a gestão de risco do negócio como um todo. Tudo isto só é possível através de um volume exponencial de dados que contribui para que a organização tenha em sua posse toda a informação necessária para tomar de uma forma mais certeira e convicta cada decisão estratégica.

Ora, tal só é possível se, de facto, existirem processos e ferramentas que auxiliem a tomada de decisão, tendo em conta que o volume de dados é tão grande que seria impossível interpretá-lo sem recurso a softwares de gestão de informação. Gera-se então este enorme volume de dados, de diferentes origens e em diferentes formatos, de tal forma que o *Big Data* se torna também uma preocupação das empresas, com o intuito de extrair o que é realmente importante de toda essa informação.

A distinção entre *Business Intelligence* e *Big Data* é ténue e estes dois conceitos, embora possam existir separadamente, complementam-se e contribuem para que as empresas possam analisar em detalhe os gostos, preferências e comportamentos dos seus consumidores, de modo a que as suas decisões estratégicas sejam um reflexo disso.

Quando se fala de empresas que atuam no setor do retalho, esta procura por uma vantagem competitiva torna-se ainda mais evidente.

Esta dissertação aborda um exemplo em contexto empresarial e trata-se de um estudo de caso aplicado numa das maiores organizações no mundo do retalho, a empresa Sonae MC, uma das empresas do grupo empresarial multinacional Sonae.

O estudo surgiu da necessidade de criar processos em ferramentas de Big Data que colmatassem algumas necessidades das áreas comerciais, dado que estas procuravam um acompanhamento mais frequente dos ganhos e perdas das suas relações comerciais.

Posto isto, foi pedido à BIT (Business Information Technology) o desenvolvimento e implementação desses processos.

Sendo uma empresa de referência no mercado nacional (e internacional) a Sonae MC procura, desde muito cedo, emancipar-se na utilização de tecnologias avançadas que permitam antecipar decisões estratégicas o mais cedo possível e, deste modo, garantir a vantagem competitiva no mercado.

1.3. OBJETIVOS

O objetivo principal deste estudo consistiu na implementação de processos de BI em ferramentas de Big Data, de forma a comprovar a sua mais-valia no auxílio à tomada de decisão. Neste sentido, os processos a implementar devem:

- Possuir, sustentar e efetuar a distribuição quer de receitas marginais quer de receitas de investimento promocionais;
- Possuir um bom desempenho;
- Cumprir os requisitos identificados e especificados;
- Criar e aumentar valor à organização.

1.4. ORGANIZAÇÃO DO RELATÓRIO

A estrutura desta dissertação passa, inicialmente, por uma revisão bibliográfica dos conceitos relacionados com o tema em estudo. De seguida, far-se-á uma exposição do problema, a sua contextualização e solução proposta. Por último, elenca-se a implementação da solução encontrada, de modo a satisfazer o propósito desta investigação.

Assim, no Capítulo 2 deste documento, é apresentado o conceito de *Business Intelligence* (BI) e de noções com este relacionadas, nomeadamente: *data warehousing*, modelação multidimensional, processos ETL e *On-Line Analytical Processing*.

O capítulo 3 apresenta a caracterização e as ferramentas de *Big Data*. Aqui, far-se-á uma distinção e comparação entre os conceitos de BI e *Big Data*.

No capítulo 4 encontra-se a contextualização e especificação do problema, bem como uma proposta de solução a implementar.

No 5º e penúltimo capítulo, encontra-se descrita a implementação realizada para responder às necessidades identificadas no capítulo anterior, bem como a apresentação dos resultados obtidos.

Por fim, o capítulo 6 aborda as conclusões alcançadas com o estudo, bem como enumera algumas limitações de investigação e elenca possíveis questões de investigação futura.

2. BUSINESS INTELLIGENCE

Neste capítulo são descritos os principais conceitos sobre *Business Intelligence* (BI).

No que diz respeito a BI, abordar-se-á três, e provavelmente os mais importantes, conceitos relacionados com este tipo de sistemas: *Data Warehousing*, modelação multidimensional e processos ETL.

2.1. INTRODUÇÃO

Cada vez mais se ouve o termo *Business Intelligence* relacionado com o mundo empresarial. Os responsáveis pela tomada de decisão sabem que possuir mais e melhor informação, atempadamente, pode diferenciar as suas organizações das demais. Para tal, o *Business Intelligence* requer soluções para recolher e estruturar dados e, posteriormente, convertê-los em informação fidedigna. Resumidamente, o *Business Intelligence* converte uma grande quantidade de dados gerados pelas empresas e apresenta-os de uma forma significativa e capaz de gerar valor acrescentado.

As áreas da gestão de conhecimento e de *Business Intelligence*, são as que têm mais importância nas discussões das organizações, pois são vitais na melhoria da informação quer na quantidade, quer na qualidade. É de frisar que a gestão de conhecimento é um processo

que conjuga o domínio humano e os objetos do domínio da informação e dos dados tendo sempre em vista a criação de valor.

Por sua vez, o objetivo principal dos sistemas de *Business Intelligence* é a melhoria da disponibilização da informação, bem como a melhoria do valor da mesma, de forma a auxiliar a tomada de decisão de uma maneira mais significativa e importante (Cody, Kreulen et al. 2002).

Segundo Alter (2002), a denominação de sistemas de BI é uma designação moderna e sofisticada, uma vez que este termo tem substituído uma designação com mais de quatro décadas, de sistemas com a finalidade semelhante – os Sistemas de Suporte à Decisão.

Em suma, poder-se-á dizer que os sistemas de BI tentam responder às mais variadas e distintas tarefas, principalmente a realização de previsões – baseadas no passado, na atualidade e até mesmo na concorrência – que se revela uma mais importantes tarefas deste tipo de sistemas. Logicamente, que a tarefa referida anteriormente, aliada à permissão de acesso *ad hoc* aos dados e à análise detalhada da organização, são o conjunto de tarefas mais comuns neste tipo de sistemas.

2.2. DATA WAREHOUSING

É cada vez mais importante que cada organização tenha noção da sua evolução. Para isto, é necessário possuir um local onde esteja alocada essa informação, ou seja, um *Data Warehouse* (DW).

Um *Data Warehouse* pode ser considerado um repositório que permite armazenar dados e, mais que isso, permite a quem lhe acede a análise dos mesmos, da forma que se julgar eficaz para a informação que se pretende obter. Tal é possível, uma vez que um *Data Warehouse* é dimensionado e criado para consolidar a informação de uma forma válida, consistente e útil.

Existe uma relação de redução de carga e trabalho entre as bases de dados operacionais e um *Data Warehouse*. Toda a informação existente num DW, possui um marco temporal, permitindo aos seus utilizadores obterem informação mais complexa. Com isto, as bases de dados operacionais podem concentrar os seus recursos nas tarefas de recolha, armazenamento e manipulação de dados (Inmon 2002).

De forma a compreender o que distingue as bases de dados operacionais de um *Data Warehouse*, a tabela 1 possui as características de cada destes tipos de repositórios.

Tabela 1 - Características das bases de dados operacionais vs *Data Warehouses*. Elaboração própria, com base em revisão de literatura.

Base de dados operacionais	**Data Warehouses**
• Foco operacional	• Registo histórico
• Acessos de leitura/escrita	• Acesso só de leitura
• Transações predefinidas	• Relatórios periódicos
• Acesso a poucos registos de cada vez	• Acesso a muitos registos de cada vez
• Dados atualizados em tempo real	• Carregamentos periódicos de mais dados
• Estrutura otimizada para atualizações	• Estrutura otimizada para processamento de *queries*

Segundo Gardner (1998), um dos grandes desafios na criação de um *Data Warehouse* é distinguir dados operacionais de dados informativos. Por isso, a informação deve corresponder de forma quase personalizada a cada unidade da organização e suas necessidades.

2.3. CARACTERIZAÇÃO

Um *Data Warehouse* é uma base de dados especificamente projetada e alimentada para dar suporte à tomada de decisões numa organização. Trata-se então de uma base de dados que contém grandes volumes de dados, sendo a sua capacidade de armazenamento uma das suas valiosas características.

Ralph Kimball (Kimball 1996) afirma que "*a data warehouse is a copy of transaction data specifically structured for query and analysis*", ou seja, um DW é uma réplica dos dados transformados e estruturados para servir consultas e análises.

Um dos pioneiros nesta temática, William H. Inmon (Inmon 1996), vai mais além, definindo que "*a data warehouse is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management's decision-making process*". Desta forma, um DW armazena conjuntos de dados: (1) orientados por assunto, (2) integrados, (3) catalogados temporalmente e (4) não voláteis, auxiliando assim a tomada de decisão.

Em suma, a primeira característica identificada por William H.Inmon – a orientação por assunto – significa que um DW organiza os dados consoante a sua relação com o negócio[1] ou atividade organizacional. Alguns exemplos de organização de dados por assunto são, nomeadamente, organizar os dados por clientes, funcionários ou fornecedores (Inmon 1996).

A segunda característica significa que os dados são armazenados num formato consistente, através do uso de convenções de nomenclatura, restrições de domínio, atributos físicos e métricas (Inmon 1996).

Ao associar um marco temporal aos dados permite a um DW possuir a características de catalogação. Por sua vez, a não volatilidade, quarta e última característica, significa que os dados permanecem inalteráveis após a sua integração num *Data Warehouse* (Inmon 1996).

Ao analisar estas características, um DW pode-se descrever como um repositório de dados robusto, que sustenta um modelo de dados de apoio a tomada de decisão (Han and Kamber 2001).

Ao dimensionar um *Data Warehouse*, deve-se ter em consideração os diferentes domínios de informação. Algumas organizações possuem um DW específico com uma estrutura de dados adaptada a cada domínio de dados. Com isto, as análises e, consequentemente, a tomada de decisão torna-se mais rápida e eficaz. Este tipo de repositório designa-se por *Data Mart*.

Na figura 3, pode-se verificar alguns exemplos de arquitetura que cada organização poderá adotar, tendo por base única e exclusivamente um *Data Warehouses* ou *Data Mart's*, ou a junção entre os dois, uma vez que estes podem coexistir.

[1] O termo negócio é muito utilizado e associado quer ao conceito de *Business Intelligence* quer a *Data Warehousing*. Não se deve associar este termo única exclusivamente a atividades que se possam traduzir em lucros, mas a todas e quaisquer atividades de uma determinada organização.

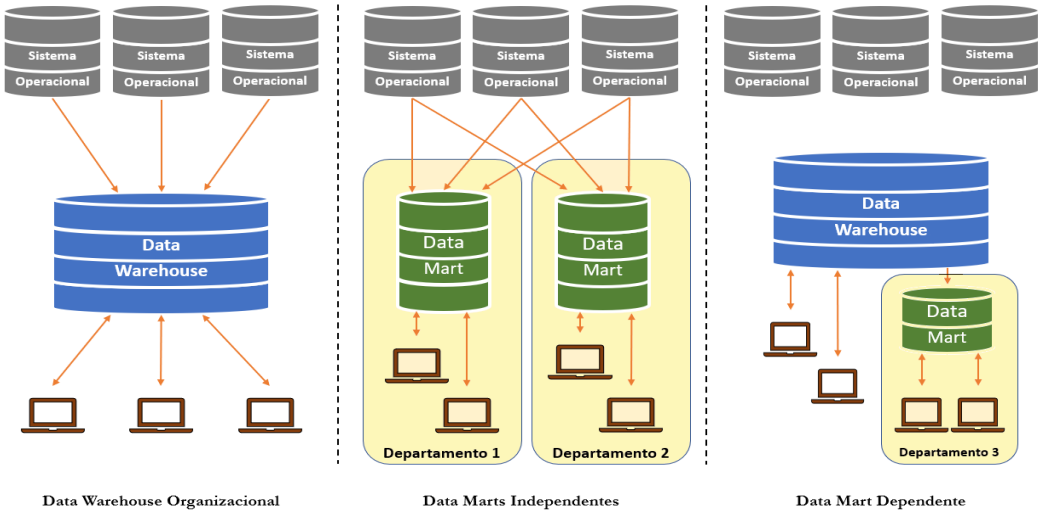


Figura 3 - Diferentes arquiteturas de implementação de *Data Warehouses* e *Data Marts*. Elaboração própria, com base em Gardner (1998).

Em síntese, um *Data Mart* obedece às mesmas características que um *Data Warehouse*. Posto isto, ambos têm como principal e fulcral objetivo armazenar dados estruturados, tentando oferecer um melhor apoio a quem toma decisões no ceio das organizações.

2.4. MODELAÇÃO MULTIDIMENSIONAL

O conceito de modelação multidimensional consiste em perceber que informação se pretende apresentar e, partindo deste pressuposto, construir ligações entre os diferentes dados. Para melhor compreensão desta temática, é importante ter-se presente o conceito e designação do que é uma tabela de factos e uma tabela de dimensão.

As tabelas de factos são tabelas que armazenam os acontecimentos, transações ou elementos associados a um determinado processo de negócio. Cada linha corresponde a um evento. Estas tabelas guardam o que acontece numa determinada organização, estando nelas a informação que se pretende analisar. Importa frisar que, todos os acontecimentos que partilhem a mesma tabela factual devem possuir a mesma granularidade, ou seja, o mesmo nível de detalhe.

"A dimensional model distinguishes between facts and attributes. A fact is usually something that is not known in advance. A fact is "*an observation in the marketplace.*" (Kimball, Reeves et al. 1998).

As tabelas de dimensão armazenam registos que completam e acrescentam informação às tabelas de factos. Nestas tabelas encontra-se mais contextualização do que aconteceu num facto, pois geralmente são dados de referência que se encontram nas tabelas de dimensão. Estes tipos de tabelas costumam possuir uma chave primária, normalmente designada pela sigla PK, e uma chave natural, designada pela sigla NK. Estas chaves naturais normalmente são provenientes dos sistemas operacionais. Importa sublinhar que existe também a sigla SK e representa o conceito de *surrogate key*, que está relacionada diretamente com a chave primária, dado que esta SK é uma substituta da chave primária e é criada para ser a chave da dimensão.

Nas tabelas de factos podem-se encontrar factos aditivos, factos semi-aditivos e factos não aditivos. De uma forma genérica, os factos aditivos são factos que podem ser agregados por todas as dimensões que estão presentes no modelo. Os factos semi-aditivos consistem nos factos que podem ser agregados apenas por uma parte das tabelas de dimensão existentes no modelo. Por fim, os factos não aditivos representam os que não podem ser agregados, pois a agregação deste tipo de factos não traduz a realidade dos acontecimentos que se pretendem analisar.

Associados a tabelas de dimensão estão os conceitos de *Fast Changing Dimension* e de *Slowly Changing Dime*nsion (SCD). O primeiro conceito surge em tabelas de dimensão cuja informação tem grandes alterações ao longo do tempo, ou seja, os dados estão em constante mutação. Por sua vez, o conceito de *Slowly Changing Dimension*, traduz exatamente o contrário, pelo que os conteúdos das dimensões não mudam constantemente.

Kimball e Ross (2013), identificaram estratégias para concretizar alterações de atributos. No anexo I estão identificados as estratégias e os impactos que cada tipo de alteração pode causar num modelo multidimensional.

Após esta explicação dos componentes básicos intervenientes na modelação multidimensional, pode constatar-se que este conceito é o responsável na temática de *Data Warehousing*, por idealizar a estrutura de sistemas a implementar. Os pilares da modelação multidimensional são simples: criar estruturas de dados fáceis de compreender e utilizar; e a otimização do desempenho no processamento.

A concretização dos pressupostos identificados é alcançada através da implementação de esquemas multidimensionais. Os tipos de esquema que podem ser implementados são: esquema em estrela, floco de neve e constelação.

O esquema em estrela é o mais utilizado na modelação multidimensional, sendo por isso a forma mais comum de modelar dados (Moody and Kortink 2003). Este esquema tem como intuito facilitar a consulta a um DW. Tipicamente, este tipo de esquema possui uma única tabela de factos ligada a várias tabelas de dimensão. Graficamente, um esquema em estrela é desenhado com uma tabela factual no centro e as respetivas tabelas de dimensão ligadas em torno, tal como se pode constatar na figura 4.

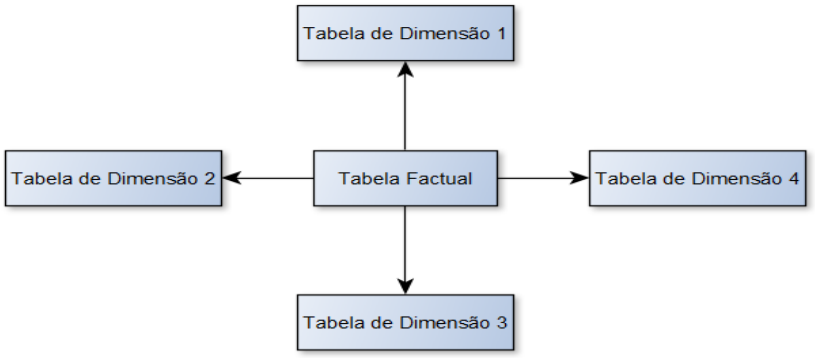


Figura 4 - Esquema em Estrela. Elaboração própria.

A ligação entre tabela de dimensão e as tabelas de factos pode definir-se como uma relação de um para muitos (1:n), como se pode verificar na figura 5. Esta relação acontece quando diferentes factos possuem o mesmo detalhe.

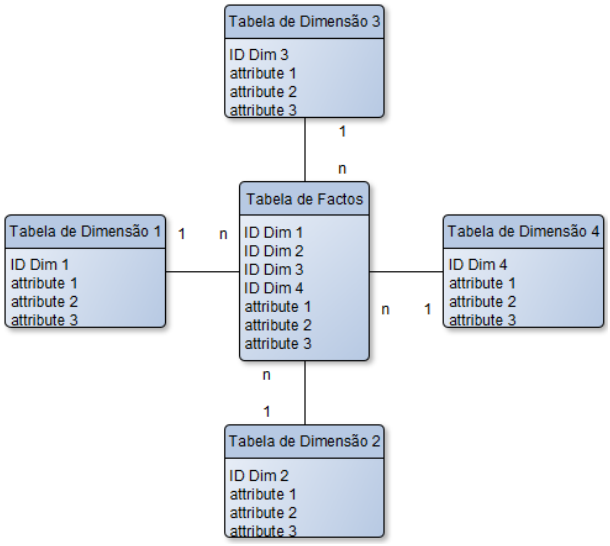


Figura 5 - Modelo Relacional de um esquema em estrela. Elaboração própria.

No que diz respeito ao esquema em floco de neve, pode-se constatar que existem algumas diferenças entre este e o esquema apresentado anteriormente. A maior vantagem existente neste tipo de esquema é a clareza da estrutura das tabelas de dimensão, ao contrário do esquema em estrela (Moody and Kortink 2003). Importa referir que no esquema em floco de neve, geralmente, as dimensões encontram-se parcial ou completamente normalizadas, o que evita a ocorrência de redundância.

Se, por um lado, se tem algumas vantagens na utilização deste tipo de esquema, por outro pode-se afirmar que as vantagens do modelo em estrela continuam intocáveis por este modelo, dada a elevada complexidade de interpretação e a perda de desempenho que o esquema floco de neve apresenta, devido ao processo de normalização das tabelas de dimensão (Han and Kamber 2001). Na figura 6 pode-se visualizar um exemplo de esquema em floco de neve.

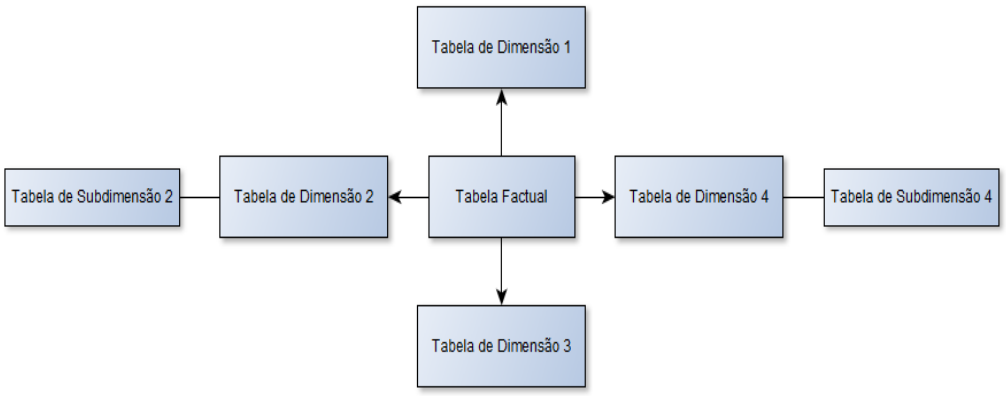


Figura 6 - Esquema em Floco de Neve. Elaboração própria.

Por sua vez, o esquema em constelação pode ser classificado como a junção de diversos esquemas em estrela. Esta designação deve-se ao motivo de, neste tipo de esquema, diferentes tabelas de factos partilharem uma ou mais tabelas de dimensão, tal como se pode verificar na figura 7.

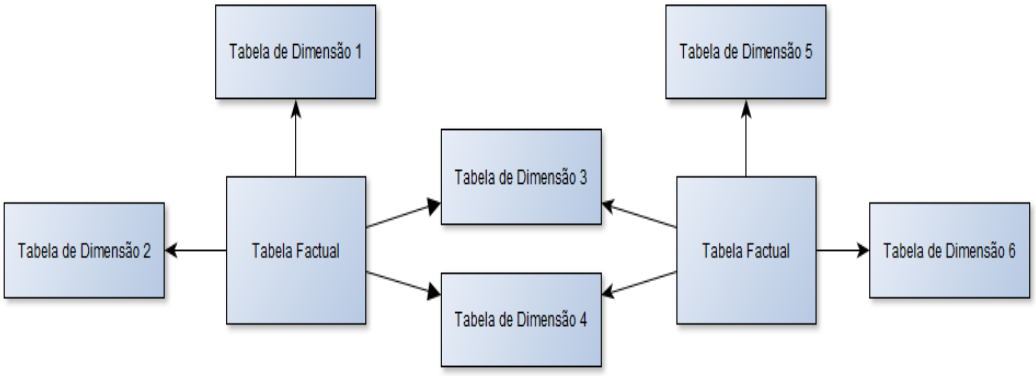


Figura 7 - Esquema em Constelação. Elaboração própria.

Assim, existem três esquemas possíveis para efetuar a modelação multidimensional. Face a não haver apenas um modelo, por vezes pode ser difícil perceber qual é o esquema mais correto a adotar em determinada situação. Para ajudar a perceber qual o esquema mais indicado, Kimball e Ross (2013) identificaram quatro etapas para ajudar essa seleção. As etapas definidas foram:

- Identificação dos processos de negócio;
- Identificação da granularidade;
- Identificação das dimensões;
- Identificação dos factos.

Na etapa de identificação dos processos de negócio deve-se ter em conta o levantamento de requisitos. Com esta identificação existe uma maior perceção das necessidades, levando a uma melhor priorização.

A priorização dos processos de negócio pode ser definida através da avaliação de dois fatores: impacto e concretização. Face à relação concretização-impacto, pode-se compreender por qual processo de negócio se inicia a implementação. Posto isto, o primeiro processo de negócio a implementar é aquele com maior impacto potencial no negócio e maior capacidade de concretização. Na figura 8 pode-se visualizar essa grelha de priorização.

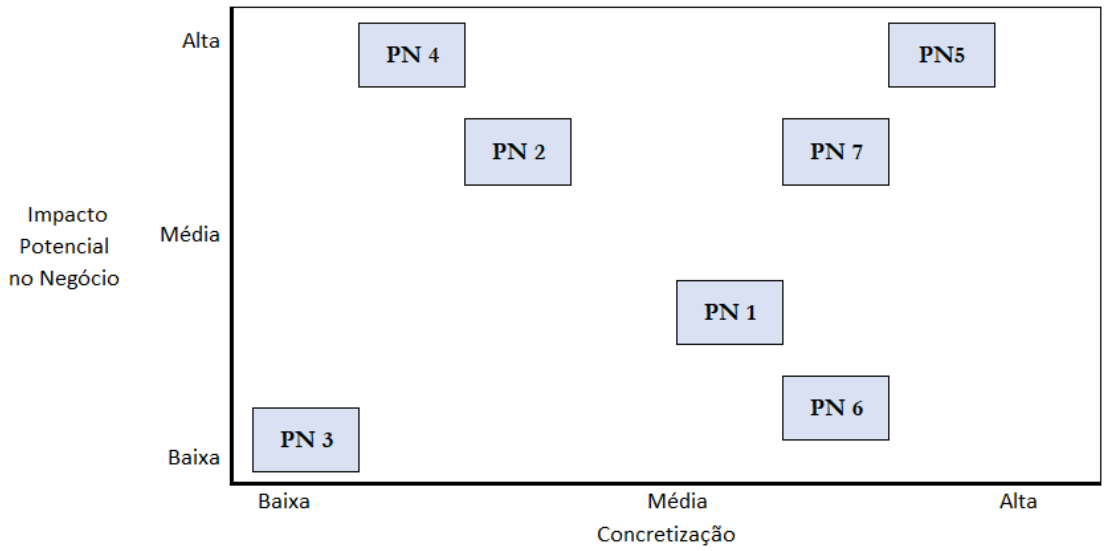


Figura 8 - Grelha de priorização dos processos de negócio. Elaboração própria, com base em Kimball e Ross (2013).

A segunda etapa, para identificação de granularidade, destina-se a perceber a que nível o detalhe será apresentado nas tabelas que compõem o modelo de dados. Quando se aborda granularidade trata-se, por exemplo, de definir se uma determinada tabela apresentará os dados ao dia, mês ou ano. Esta determinação é importante para a implementação dos processos, pois influencia diretamente os desenvolvimentos e o conteúdo das tabelas.

A terceira e quarta etapas descrevem de forma quase imediata o seu intuito. A etapa de identificação de dimensão permite dar enquadramento das métricas em analise pois é nestas dimensões que se encontram as descrições relacionas com as tabelas factuais. Por sua vez, a identificação dos factos indica o que efetivamente se pretende analisar, pelo que esta etapa não pode ser desconsiderada. É importante frisar que nesta quarta e última etapa os factos identificados devem possuir a granularidade definida na etapa com esse fim.

Como já referido anteriormente, as necessidades do negócio vão surgindo ao longo do desenvolvimento e crescimento de uma determinada organização. Qualquer modelo multidimensional deve possuir capacidade de ajuste e alteração. Geralmente, as alterações solicitadas mais comuns são: a adição de atributos às dimensões; a identificação e consequente criação de novas dimensões; e a identificação e criação de novos factos.

Em suma, existe um conjunto de ações que levam à perceção de qual a modelação mais correta de forma a colmatar as necessidades e requisitos identificados, apresentando assim um modelo completo, robusto, fácil e adaptável.

2.5. PROCESSO ETL

O termo ETL advém dos termos *Extraction, Transformation, Loading* que, tal como estes termos indicam, estes processos são os responsáveis por tratar os dados, de um ponto de vista muito simplista. É através destes processos que é possível homogeneizar, limpar e carregar os dados para um *Data Warehouse* (Vassiliadis, Simitsis et al. 2002).

Para se conseguir carregar os dados de forma clara e organizada às necessidades do negócio, é necessário ter um conjunto de processos que permitam aceder aos dados na sua origem, tratá-los e transformá-los, entre outras tarefas necessárias para efetuar o carregamento dos mesmos num determinado *Data Warehouse*. Neste tipo de processo deve-se possuir uma área de trabalho e uma estrutura de dados bem definida para que os carregamentos aconteçam com o menor ruído possível. Pode-se então constatar que as ferramentas de ETL são os conetores de dois mundos, o sistema operacional e o *Data Warehouse* (Kimball and Ross 2013).

Um processo ETL está divido em três etapas, que consistem na extração, transformação e carregamento de dados. Na primeira etapa, o objetivo é mover os dados de uma determinada fonte para uma área de *staging*. Nesta primeira etapa não é muito comum efetuar transformações com grande impacto nos dados, no entanto, pode-se realizar já pequenas ações, nomeadamente a verificação dos tipos de dados que se está a extrair.

A segunda etapa, de transformação, pode ser designada pela etapa mais morosa de um processo ETL. Nesta etapa todas as grandes mutações aos dados acontecem através de uma série de rotinas de limpeza e transformação. A verificação da qualidade e integridade dos dados bem como a adoção de ações corretivas sobre os mesmos, podem ser rotinas integrantes desta etapa. Geralmente, esta etapa tem as suas tarefas definidas, visando um formato final de dados que deverá ter em vista a inserção dos dados sem mais nenhuma alteração. Estas tarefas normalmente são executadas numa outra área de *staging*.

A terceira e última etapa, que consiste no carregamento, não possui por norma muitas tarefas, ficando a seu cargo a inserção no *Data Warehouse* dos dados prontos para análise e consulta. Pode afirmar-se que a etapa de carregamento não diz respeito exclusivamente a carregamento de novos dados, mas também à atualização de dados já existentes nas tabelas do modelo. Na figura 9 pode-se verificar estas três etapas de um processo ETL.

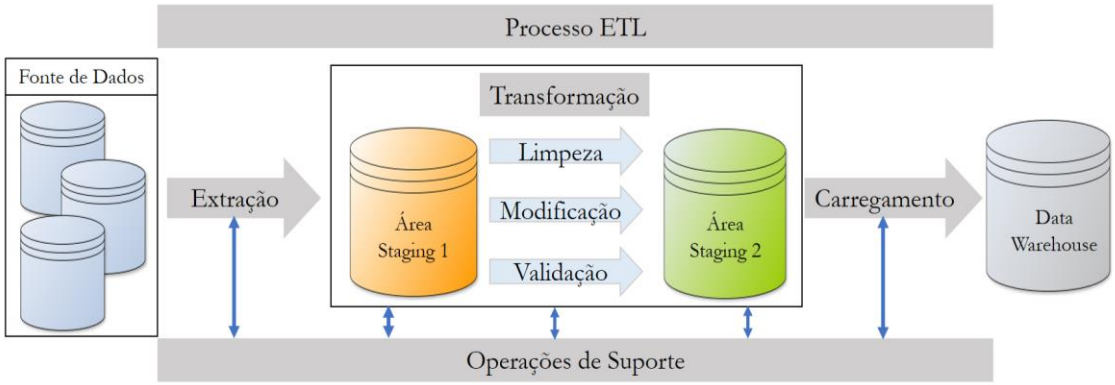


Figura 9 - Processo ETL – Fluxo dos dados. Elaboração própria, com base em Kimball e Caserta (2004).

Uma vez que este processo é muito orientado para obter uma determinada estrutura de dados, muitas vezes é concebido do fim para o início. Com esta técnica de especificação, todas as ações a realizar no processo são projetadas e desenvolvidas com o propósito de alcançar o *output* desejado. Com isto, somente os dados necessários para criar o formato de dados que se pretende são incluídos na etapa de extração.

A implementação de um processo ETL, consome geralmente bastante tempo na sua realização. Kimball e Caserta (2004) destacam que as tarefas associadas a um processo ETL chegam a consumir cerca de 70% dos recursos necessários para a implementação e manutenção de um DW.

A implementação de um processo ETL passa por várias análises, porém importa ressalvar que existem alguns que merecem um lugar de destaque. Alguns dos requisitos que devem ser tidos em conta análise de pré-implementação deste tipo de processos são: os requisitos de negócio; ser *compliance* (os dados que se disponibiliza estarem de acordo com requisitos legais, se existirem); a qualidade e integração os dados; a latência (rapidez de disponibilização dos dados que estão na fonte ao suporte de tomada de decisão) e licenças.

Na figura 10, constata-se um gráfico que traduz a relação entre a velocidade de um processo ETL e a qualidade de dados.

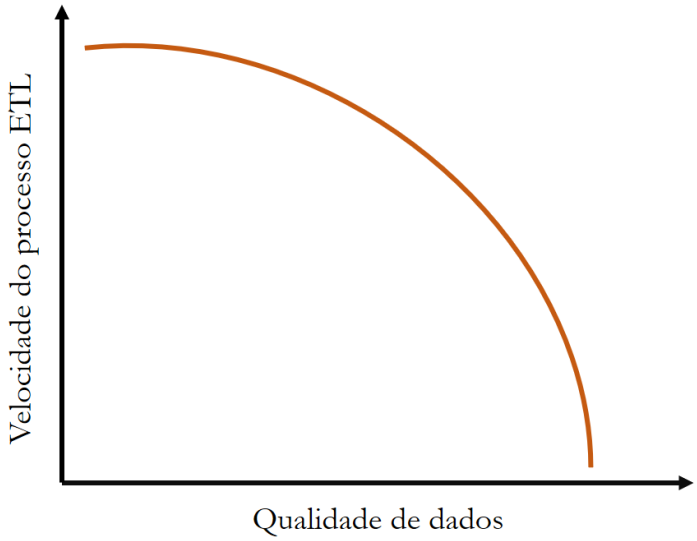


Figura 10 - Gráfico de relação entre velocidade do processo ETL e a qualidade dos dados. Elaboração própria, adaptado de Kimball e Caserta (2004).

Em suma, um processo ETL ocupa um grau de importância enorme quando se fala nos conceitos de *Data Warehousing* e *Business Intelligence*. Além disso, o objetivo de um processo deste tipo é alimentar uma estrutura de dados final, cumprindo assim os requisitos definidos.

3. BIG DATA

Big Data é a designação utilizada para uma situação onde se pretende obter, processar, armazenar e analisar grandes volumes de dados, que não podem ser processados utilizando ferramentas convencionais de processamento de dados.

Segundo Akerkar (2014), *Big Data* é uma terminologia usada para descrever grandes volumes de dados, sendo estes estruturados ou não, por organizações que querem obter valor após a análise da informação.

Tanto em âmbitos académicos como profissionais, cada vez mais se realça a necessidade de compreender se as aplicações de análises de dados em *Big Data* podem (ou não) ser consideradas uma mais valia para as empresas alcançarem uma vantagem competitiva (Côrte-Real, Oliveira et al. 2017).

A definição do termo *Big Data* não é unânime pelos diferentes autores científicos. Maitrey e Jha (2015) aferem que *Big Data* aborda um volume elevado de informações, que necessitam de ferramentas específicas, de forma a serem analisados. Por sua vez, YU et al. (2015) definem *Big Data* como grandes volumes de informação e dados e, mais ainda, alegam que *Big Data* dispõe de uma característica que torna impossível uma análise numa ferramenta analítica comum, devendo esse volume ser analisado num ambiente adequado. Os autores afirmam também que, uma vez ultrapassada essa barreira de perceber onde e como tratar e analisar os dados, se obtêm melhorias no processo de tomada de decisão.

Hal Varian e Peter Lyman, da Universidade da Califórnia em Berkeley, durante três anos (2000-2003), conduziram pesquisas sobre o volume de dados produzidos, armazenados e transmitidos, por organizações, governos e pessoas, designado por "*How much Information*?". Esse estudo realçou o facto de que ocorreu um aumento na ordem dos 25% no volume dos dados gerados e armazenados em todo o mundo, no período compreendido entre o ano de 1999 ao ano de 2002 (Manyika 2011).

Segundo Reynolds (Reynolds 2016), o propósito e valor do *Big Data*, baseia-se na capacidade de oferecer informação válida a quem toma as decisões e define o caminho de uma organização.

A IBM (*International Business Machine*), frisou que 90% dos dados mundiais foram produzidos nos últimos anos. Nesta temática, os dados podem ser gerados em diferentes formatos e de diversas origens, contudo, os dados gerados nos tempos mais recentes tem maioritariamente origem de *smarthphones*, redes sociais e *e-commerce*.

3.1. CARATERIZAÇÃO

A velocidade, o volume e a variedade de dados são claramente as três características que mais se destacam para caraterizar e definir os cenários de *Big Data* atuais. No entanto, alguns autores defendem que existem mais duas características emergentes, como o valor e veracidade. Esta caracterização pode designar-se de 5 V's do *Big Data*, tal como se pode verificar na figura 11.

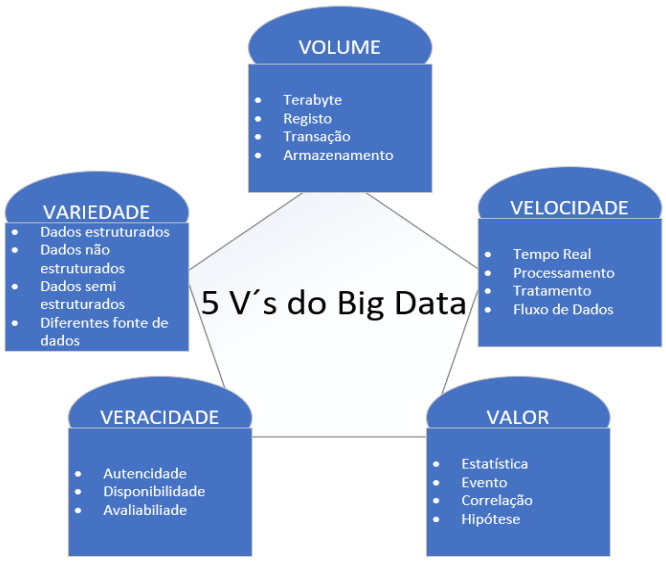


Figura 11 - Os 5 V's do *Big Data*. Adaptado de Ishwarappa & Anuradha (2015).

O primeiro V dos cinco e provavelmente o mais fulcral nesta temática é o volume. Esta característica está ligada à enorme quantidade de dados produzidos e envolvidos nos processos mundiais. De acordo com a *International Data Corporation* (IDC) em 2011, a informação mundial duplica a cada dois anos. Com isto estima-se que até 2020 exista cerca de 35 Zettabytes (1 Zettabyte = 1 Bilião de Terabytes) de dados armazenados no mundo.

Relativamente à velocidade, esta característica dos 5 V's é definida de acordo com o quão rápido os dados são extraídos, armazenados e recuperados. Esta característica está relacionada quer com a taxa de fluxo de dados, quer com a taxa de criação de dados. Assim, devido à velocidade enorme com que os dados são gerados e transmitidos, os sistemas tradicionais de análise podem não conseguirem manipulá-los nem os armazenar.

Quando se fala de variedade pode-se constatar que os formatos ou tipos de dados armazenados pelas organizações são inúmeros, e podem ser desde um tipo de dado existente numa base de dados, documentos das organizações e estruturas, notícias, entre outras fontes.

Os dados em *Big Data* podem provir de diversas fontes, estruturadas, semiestruturadas e não estruturadas. Os dados estruturados são dados armazenados em bases de dados tradicionais normalmente organizados em tabelas. Por sua vez, os dados semiestruturados são dados que podem seguir diversos padrões de forma heterogénea. Por fim, os não estruturados são dados provenientes de várias fontes distintas, como vídeo, texto, áudio, imagens, XML (*eXtensible Markup Language*), entre outros.

É de referir também que os sistemas tradicionais não conseguem armazenar, processar e interpretar esta variedade de dados. É precisamente por isso que se deve utilizar novas tecnologias, algoritmos, e técnicas para realizar a análise desses dados, tanto estruturados quanto não estruturados, em conjunto.

Para vários autores, a variedade dos dados é resultante da forma como se consegue obter a informação, por exemplo, através de uma publicação nas redes sociais, de uma imagem, de dados resultantes de um sensor. Para Dumbill (2012), nenhuma destas origens está pronta para integração numa aplicação.

Outra característica do *Big Data* é a veracidade. Esta característica torna-se importante pois alguns autores defendem que a precisão e a confiança das análises dependem da

fiabilidade dos dados na sua origem. Com isto, pode-se afirmar que a experiência de utilização de uma determinada fonte de dados leva com que a confiança e crédito dos analistas aumente sobre a qualidade dos dados (Ramachandramurthy, Subramaniam et al. 2015).

Em suma, a veracidade é a confiabilidade dos dados. Os dados armazenados devem ser confiáveis, realistas e não inventados. A qualidade e consistência dos dados são características que se deve ter em consideração ao armazenar dados para análises em *Big Data*. Além disso, importa ressalvar que dados gerados internamente nas organizações são validados mais facilmente que os gerados externamente.

A última característica, mas não menos importante, é o valor. Esta característica está diretamente relacionada com a capacidade de desenvolver processos que tornem um conjunto de dados estruturados (ou não) em informação limpa, organizada, útil, capaz de acrescentar valor às estruturas das organizações. Elisabeth et al. (Elisabeth, Anikó et al. 2015) defendem que, se for possível extrair algo útil da diferente informação que é gerada, poder-se-á obter resultados quer a nível económico, quer social, de relevo para as estruturas e organizações.

Existem ainda poucos autores que defendem outras características. Entre muitas, destacam-se as inúmeras formas de demonstrar os resultados obtidos após as análises de dados e a capacidade de os sistemas deverem ser, cada vez mais, escaláveis e mais eficientes, tal como defendem Chen, Wu e Wang (2015). Estas caraterísticas designam-se de visualização e escalabilidade, respetivamente.

3.2. FERRAMENTAS DE BIG DATA

Com a evolução dos tempos e, consequentemente, com a evolução dos cenários de *Big Data*, foi necessário o aparecimento de novas ferramentas, com características que fossem ao encontro dos requisitos necessários para trabalhar grandes volumes de dados.

Quando se fala de *Big Data*, fala-se obrigatoriamente de paralelismo, ou seja, a capacidade que uma ferramenta tem em processar quantidades de dados menores ao mesmo tempo em diferentes máquinas, diminuindo assim a carga computacional em cada máquina, podendo até exigir-se menos capacidade de processamento a cada máquina.

Neste tipo de ferramentas encontra-se também uma característica que permite de forma quase personalizável, a cada necessidade existente nas organizações, ter infraestruturas que se conseguem adaptar ou estar aptas a aumentar a sua capacidade, de forma a corresponder de forma mais eficiente às necessidades existentes.

Uma arquitetura de *Big Data* pode incluir diferentes infraestruturas, ferramentas, técnicas e linguagem de programação implementadas. Na figura 12, pode ser visto um resumo de um ecossistema de *Hadoop*, uma das ferramentas de *Big Data* possíveis.

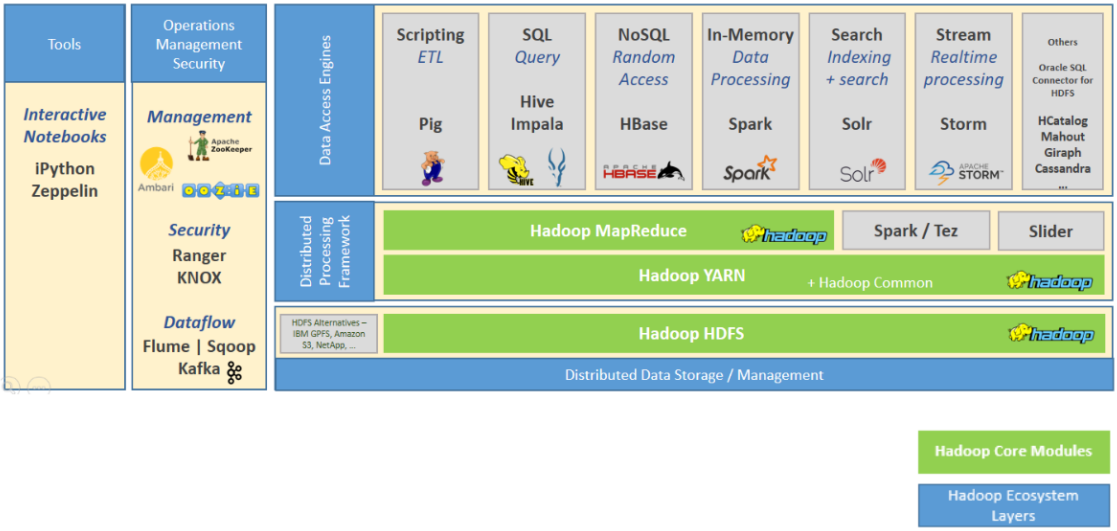


Figura 12 - Ecossistema *Hadoop (Jain 2017).*

Em suma, existem várias tecnologias, técnicas, ferramentas e até mesmo linguagens de programação que se pode utilizar nos diferentes cenários do *Big Data*. No entanto, não se pode afirmar que uma é superior a outra pois cada caso é um caso. Deve-se sim, analisar as necessidades de processamento e desafios existentes nas organizações e procurar encontrar. a solução que melhor responde às mesmas. No entanto, no âmbito desta dissertação, destaca-se o *Hadoop* e o *Spark*.

3.3. HADOOP

O *Hadoop* é um projeto desenvolvido e mantido pela *Apache System Foundation,* que surgiu no ano de 2006. Esta solução tem como principal característica a sua capacidade de processamento de dados em larga escala, tendo uma infraestrutura computacional

distribuída. A arquitetura de dados encontrada para garantir essa capacidade de armazenamento foi a utilização de uma arquitetura de *clusters*.

Um *cluster* é um conjunto de servidores que funcionam como um sistema unificado, permitindo:

- Alta disponibilidade;
- Balanceamento de carga;
- Processamento em paralelo.

Devido a esta arquitetura em *cluster*, o *Hadoop* permite uma escalabilidade horizontal em vez de vertical, ou seja, permite acrescentar o número de servidores no *cluster* sem que se tenha que fazer um forte investimento na estrutura com novo hardware. Assim, o *Hadoop* permite acrescentar servidores à medida que as necessidades vão também aparecendo nas organizações.

O *Hadoop* encontra-se associado a três noções:

- HDFS (*Hadoop Distributed File System*);
- *Yarn*;
- *MapReduce*;

O HDFS é o conceito que permite processar volumes de dados gigantes, partindo-os em pequenos grupos de dados e distribuindo-os pelos diferentes *data nodes*. Geralmente, esses blocos mais pequenos de dados são de 64 *MegaBytes*. É através deste conceito que o *Hadoop* ganha a característica de paralelismo. Na figura 13 pode-se ver um exemplo de uma estrutura HDFS.

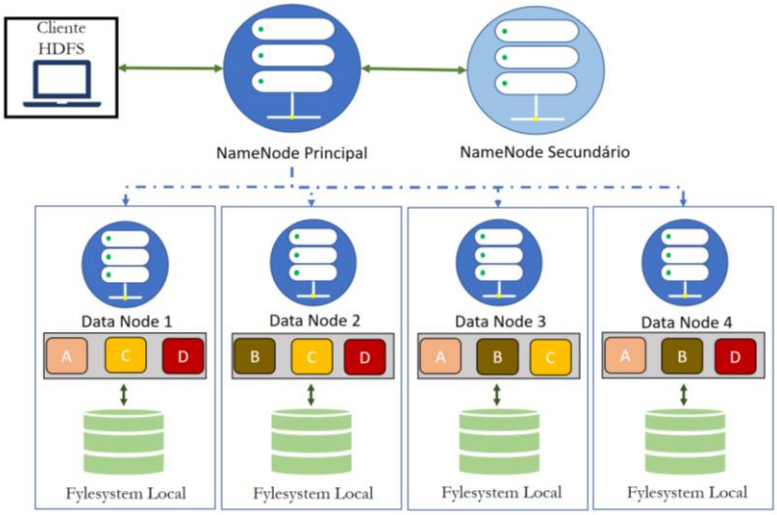


Figura 13 - *Hadoop Distributed File System*. Elaboração própria.

É de referir que o *NameNode* é o responsável pela distribuição, balanceamento de carga e consequente replicação de blocos. Está também a seu encargo a monitorização do estado dos diferentes *data nodes*, verificando se estão ou não em funcionamento. Convém ressalvar que os blocos mais pequenos de dados são guardados em *fylesystems* locais.

O outro conceito é o *Yarn*. Esta tecnologia é a responsável pela gestão do *cluster*. Através deste software o *Hadoop* é capaz de suportar uma variedade de abordagens de processamento e uma maior variedade de aplicativos. Com a ajuda do *Yarn*, os *clusters* conseguem executar processamentos e consultas interativas simultaneamente com tarefas de *MapReduce*. Na figura 14 encontra-se um diagrama que descreve o funcionamento do *Yarn*. Em suma, o *Yarn* ocupa um lugar importante pois é utlizado no processo de gestão de recursos.

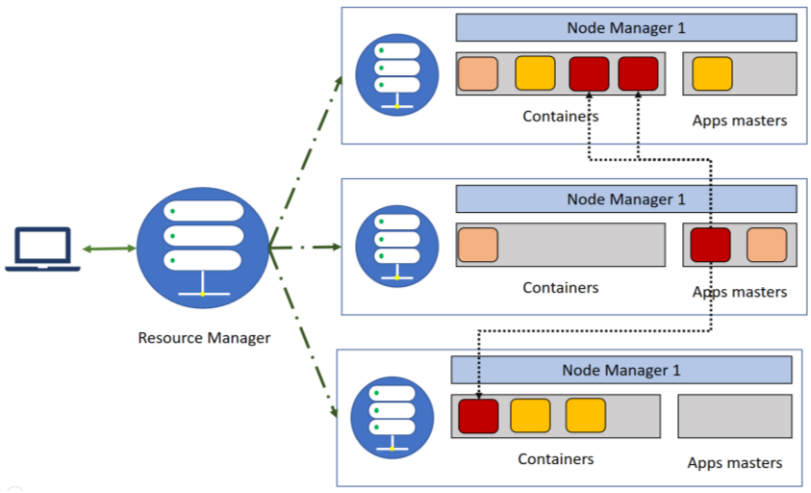


Figura 14 – Diagrama de funcionamento do *Yarn*. Elaboração própria.

O último conceito a abordar é o de *MapReduce*, que consiste num modelo de programação para processamento de dados. Consequentemente, o *MapReduce* é um processo ou tarefa que o *Hadoop* executa.

O conceito *MapReduce* pode ser subdividido em dois outros termos. As funções de "*Map*" e "*Reduce*" desempenham, portanto, um papel importante nos cenários de *Big Data* que utilizam esta solução. Com isto, as funções de "*Map*" consistem em efetuar transformações programadas, como processamentos em paralelo e alocação de recursos. Por sua vez, as funções de "*Reduce*" são responsáveis por gerar um único *output*, ou seja, acontecem depois das funções de "*Map*" e executam combinações com os dados, tendo como objetivo um *output* organizado e apresentável. Das funções de "*Reduce*" resulta a informação que havia necessidade de obter. Na figura 15 pode ser visto o funcionamento do *MapReduce*.

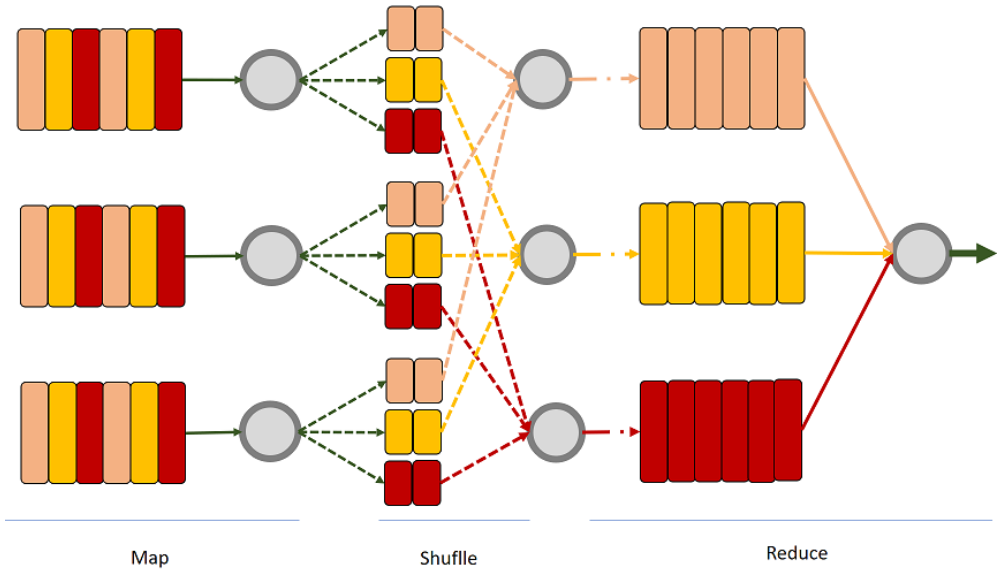


Figura 15 - Diagrama de funcionamento do *MapReduce*. Elaboração própria.

Em suma, o *MapReduce* é uma estrutura de software idealizado para processar grandes quantidades de dados em paralelo, em grandes *clusters*, de uma maneira confiável.

Por último, importa frisar que existe uma linguagem de programação muito utilizada e que é uma variante do SQL, designada por *Hive Query Language* (HiveQL abreviado ou apenas HQL).

Esta linguagem oferece a quem o utiliza um poder de abstração de tudo o que foi descrito anteriormente. O *Hive* converte a maioria das consultas em tarefas do *MapReduce*, explorando assim a escalabilidade do *Hadoop*, enquanto apresenta uma abstração similar à que o SQL oferece.

Assim, segundo Davenport (2006), o *Hadoop* é uma solução robusta, que permite adaptar a solução face ao volume, complexidade e necessidades, é um ambiente coeso de armazenamento e processamento e que permite paralelismo.

3.4. SPARK

O *Spark* foi desenvolvido em 2010, num projeto levado a cabo pela Universidade da Califórnia, em Berkeley. O *Spark* veio propor uma melhor abstração de dados e mais rapidez de execução. Além disso, tencionou também oferecer uma solução mais *user friendly* para quem desenvolve em ferramentas de *Big Data* (Arias, Gamez et al. 2016).

O intuito da criação do *Spark* foi a melhoria no processamento em paralelo, de grandes volumes. Esta necessidade surge pelo facto de o *MapReduce* não possuir um desempenho elevado se se tiver a necessidade de efetuar as operações de *map* e *reduce* várias vezes.

Este modelo de programação para processamento de dados não necessita de correr exclusivamente em *Hadoop,* contudo, também não o substitui, pois faz uso dos seus recursos nas suas execuções. Isto é, quando se pretende executar um *job Spark*, o mesmo vai pedir recurso ao *Hadoop* para realizar as operações.

Ambos os modelos de programação são eficazes e até podem coexistir. No entanto, a grande diferença entre o *Spark* e o *MapReduce* são que o primeiro modelo executa sempre as ações em memória, enquanto o segundo entre a fase de *map* e *reduce* tem que aglomerar todos os dados em disco para voltar a distribuir para as máquinas, para que a operação de *reduce* aconteça. Logo, o *Spark* executa as tarefas definidas em paralelo utilizando o máximo de memória que lhe for atribuída e disponível.

Como se pode visualizar na figura 16, o *Spark* permite construir algoritmos com encadeamento de funções. Talvez por esse motivo não oferece tantos limites como o *MapReduce*. Com isto, o *Spark* oferece um leque mais alargado de operações.

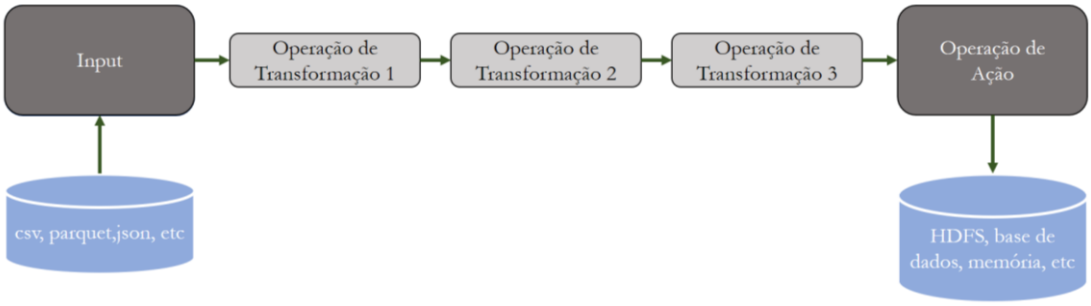


Figura 16 - Diagrama de funcionamento do *Spark*. Elaboração própria com base na revisão de literatura.

Devido a esta opção de encadeamento das operações a realizar nos dados, tendo em vista um resultado final, é possível utilizar algoritmos mais sofisticados e de forma mais simples, devido ao poder de abstração que o *Spark* oferece.

Um algoritmo idealizado em *Spark* pode ser desenvolvido em várias linguagens de programação, desde *java, phyton, scala*, entre outras. Provavelmente, a mais utilizada é a *Scala*, que consiste numa linguagem orientada a objetos. Esta pode associar-se também às *Scalable Languages*, linguagens que permitem escalabilidade, pois possuem um conjunto de requisitos que lhes concede tal característica. É designada como uma linguagem funcional onde cada função é um valor. Com isto, poder-se-á dizer que as associações entre funções são suportadas de forma natural.

Ao abordar o modelo de programação *Spark*, obrigatoriamente temos que referir o conceito de RDD (*Resilient Distributed Dataset*). Esta característica permite particionar conjuntos de dados, para efetuar processamentos em diferentes máquinas. Embora exista também noutros modelos de programação. No caso específico de *Spark*, caso alguma dessas partições seja perdida é possível recuperar. Além disso, importa referir que os conjuntos de dados criados são exclusivamente de leitura e imutáveis.

Em suma, o *Spark* é um modelo de programação para processamento de dados que permite efetuar um vasto número de operações, de uma forma simples e encadeada sobre a informação, recorrendo ao paralelismo e à capacidade de memória que lhe for alocada.

3.5. BIG DATA VS BUSINESS INTELLIGENCE

Apesar de estes dois conceitos caminharem quase lado a lado, é importante perceber e distinguir o que é *Big Data* e o que é *Business Intelligence*. De referir, que ao contrário do que muitas pessoas pensam um não é uma substituição de outro.

Conforme já explicado, são conceitos distintos e com objetivos diferentes, contudo, são complementares. As duas soluções podem coexistir numa organização aumentando até a capacidade de análise e compreensão os dados.

De forma simplista, a atenção de uma solução de BI centra-se na extração, transformação e disponibilização de dados estruturados para a tomada de decisão. Tal como já descrito, fornece indicadores (KPI's) e tendências aos gestores, para poderem criar diretrizes eficientes e eficazes para o alcance dos resultados empresariais pretendidos.

Por outro lado, o *Big Data* em geral, pouco se preocupa com a exatidão que é fornecida num sistema de BI. O *Big Data* centra-se no processamento dos dados à procura de correlações e descobertas, correlações essas que nem sempre dispõem de motivos para tal existência, pois poderá ser algo nunca analisado ou estudado.

Em suma, a grande diferença do *Big Data* é que estes tipos de soluções pretendem criar correlações desconhecidas através da análise de grandes volumes de dados, em tempo hábil, para que as empresas obtenham vantagens competitivas.

É de frisar que a implementação e desenvolvimento de uma solução de *Big Data* exige, de certa forma, um bom conhecimento em BI. Isto acontece porque normalmente as soluções de *Big Data* possuem um grau elevado de complexidade. Por conseguinte, quanto maior for a experiência em soluções que permitam familiaridade e facilitada conceção no que à análise de dados diz respeito, maior será a probabilidade de sucesso da solução.

Na tabela 2 encontra-se uma comparação sintetizada entre o conceito de *Big Data* e *Business Intelligence*.

Tabela 2 - *Big Data vs Business Intelligence.*

	Big Data	Business Intelligence
Volume de dados	Gibabyte a petabyte	Terabyte máximo
Estrutura de dados	Todos os tipos de dados	Somente dados estruturados
Custos de armazenamento	Barato por Terabyte	Relativamente caro
Accesso aos dados	Lento	Rápido
Hardware	Barato	Caro
Qualidade dos dados	Média	Alta
Novos insights	Sim	Não

Em suma, o *Business Intelligence* trata das questões conhecidas e das preconceções em relação aos dados, ao passo que *Big Data* se envolve com um universo de novas possibilidades e questões que ainda não se conhece. Com isto é possível afirmar que quer o *Big Data*, quer o *Business Intelligence*, possuem grande importância e devem ser bem entendidos para que as empresas possam aproveitá-las da melhor forma, agregando e alcançando os valores e resultados desejados aos negócios.

4. CONTEXTUALIZAÇÃO E ESPECIFICAÇÃO DO PROBLEMA

Neste capítulo encontra-se uma contextualização do problema como um todo. Apesar da solução apresentada se centrar apenas numa determinada rubrica, é importante ter a perceção que existem várias rubricas a partilhar a mesma tabela de factos e o que é uma demonstração de resultados.

Posto isto, nas três primeiras secções apresenta-se uma breve explicação do que é uma demonstração de resultados, o propósito de todos os processos, a perspetiva de todas as rubricas e o modelo multidimensional definido. Nas duas últimas secções especificam-se os processos e a lógica para as rubricas que são o foco da implementação desta dissertação.

4.1. A DEMONSTRAÇÃO DE RESULTADOS

Antes de se falar das causas e implicações dos processos desenvolvidos, importa explanar o que é uma demonstração de resultados, dado que o foco de investigação deste estudo incidiu sobre duas rubricas desta demonstração financeira.

A contabilidade tem o objetivo de disponibilizar informação de natureza económica e financeira, com o intuito de atender às necessidades dos utilizadores internos e externos à

organização. Por sua vez, a demonstração de resultados é uma das demonstrações contabilísticas produzidas e divulgadas pela contabilidade (Pires da Silva 2010).

A figura 17 ilustra as três principais demonstrações contabilísticas e a finalidade de cada uma delas.

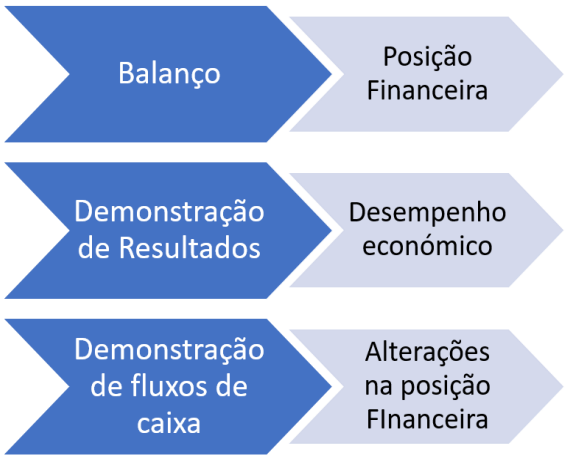


Figura 17 - As principais demonstrações contabilísticas de uma organização e seus propósitos. Elaboração própria.

O propósito da demonstração de resultados, assim como de outras demonstrações financeiras – nomeadamente, o balanço e a demonstração de fluxos de caixa – é coletar e registar dados económicos, avaliá-los em unidades monetárias e, consequentemente, sumariar a informação em relatórios para serem comunicados, de forma a que possam contribuir para a tomada de decisão.

Para tal, a informação deverá ser disponibilizada de forma estruturada dentro de um esquema de planeamento contabilístico. A tabela 3 elenca o exemplo gráfico de uma demonstração de resultados.

Tabela 3 - Exemplo gráfico de uma demonstração de resultados (Pires da Silva 2010).

RENDIMENTOS E GASTOS	Notas	Períodos	
		N	N-1
Vendas e serviços prestados			
Subsídios à exploração			
Ganhos/perdas imputados de subsidiárias, associadas e empreendimentos conjuntos			
Variação nos inventários da produção			
Trabalhos para a própria entidade			
Custo das mercadorias vendidas e das matérias consumida			
Fornecimentos e serviços externos			
Gastos com o pessoal			
Imparidade de inventários			
Imparidade de dívidas a receber			
Provisões			
Imparidade de investimentos não depreciáveis/amortizáveis			
Aumentos/reduções de justo valor			
Outros rendimentos e ganhos			
Outros gastos e perdas			
Resultados antes de depreciações, gastos de financiamento e impostos			
Gastos/reversões de depreciação e amortização			
Imparidade de investimentos depreciáveis/amortizáveis			
Resultado operacional (antes de gastos de financiamento e impostos)			
Juros e rendimentos similares obtidos			
Juros e gastos similares suportados			
Resultado antes de impostos			
Imposto sobre o rendimento do período			
Resultado líquido do período			

No que à demonstração de resultados diz respeito, o esquema contabilístico divide esta demonstração financeira entre rubricas de rendimentos e gastos. Os rendimentos, proveitos ou ganhos tratam-se de aumentos nos benefícios económicos durante o período contabilístico, ou seja, todas as operações que são suscetíveis de originar benefícios económicos presentes ou futuros para a empresa, incluindo vendas e serviços prestados, variações nos inventários de produção ou juros e rendimentos similares, entre outras rubricas possíveis (Pires da Silva 2010).

Por outro lado, os gastos consistem em todos os custos e perdas da empresa, ou seja, todas as operações que são suscetíveis de originar perdas económicas presentes ou futuros

para a empresa, incluindo custos com mercadorias e matérias-primas, custos com fornecimentos e serviços prestados por terceiros, gastos com o pessoal, imparidades, amortizações e ajustamentos do exercício.

Deste modo, a demonstração de resultados permite apurar o resultado obtido pela empresa durante o período a que este diz respeito. Se o resultado for positivo, a empresa verá a sua riqueza aumentar; se for negativo, a empresa diminuirá a sua riqueza. Assim, este mapa contabilístico torna-se um meio para analisar qual a variação da riqueza da empresa num determinado período de tempo e quais as razões dessa mesma variação.

Em suma, a demonstração de resultados reflete os recursos que a organização gerou ou consumiu durante um determinado período para a execução da sua atividade. Ao contrário do balanço, que oferece uma imagem estanque sobre a saúde financeira da organização em determinado momento, a demonstração de resultados oferece uma visão dinâmica da organização.

Além disso, a classificação dos recursos e compromissos de uma entidade em categorias apropriadas é fundamental para que os diferentes utilizadores internos e externos (investidores, instituições de crédito, administração pública, entre outros *stakeholders*) possam interpretar a informação financeira adequadamente; caso contrário, tornar-se-ia indecifrável a sua análise. Assim, a base da contabilidade é a classificação, com o intuito de agregar os dados para uma melhor interpretação dos mesmos. Neste sentido, surgem as rubricas, que funcionam como compartimentos (contas) onde se registam determinadas operações contabilísticas (por ordem de verosimilhança), de tal modo que sirvam de instrumentos para o registo sistemático de cada facto (Pires da Silva 2010).

No caso de estudo em apreço, as rubricas de receitas marginais e receitas de investimento promocional tratam-se de duas contas de rendimentos da demonstração de resultados.

4.2. O PROBLEMA

As organizações têm a necessidade de possuir cada vez mais e melhor informação sobre o seu dia-a-dia de forma a auxiliar a tomada de decisão.

Assim, os departamentos comerciais da Sonae MC sentiram necessidade de possuir indicadores de *performance* sobre o seu desempenho comercial (margem) e, com isto, possuir informações mais detalhadas e robustas, que permitissem um maior poder negocial junto dos seus fornecedores e parceiros de negócio.

Apesar de já existir informação mensal sobre as margens comerciais ao nível da categoria de produto, existia uma lacuna ao nível diário daquilo que seria a rentabilidade de um contrato comercial por dia, produto e fornecedor.

Para responder a esta necessidade, destacam-se dois problemas distintos. Primeiramente, o volume de dados a trabalhar para apresentar a informação solicitada era elevado. Por outro lado, o que desencadeou principalmente a necessidade de desenhar esta solução foi o requisito de granularidade. A granularidade máxima pedida era uma informação que, nalguns casos, nem sequer existia, o que levou à necessidade de idealizar e implementar métodos de distribuição sobre os valores quer de receitas marginais, quer de receitas de investimento promocional.

Desta forma, foi endereçado à BIT (*Business Information Tecnology*) o pedido para realização de uma nova estrutura para mitigar essas necessidades.

É de frisar que a solução anterior é designada por PLA – *Profit and Loss Analysis*, sendo que a nova estrutura foi designada **P&L Comercial**.

4.3. ÁRVORE DE RUBRICAS

Para a alteração estrutural, foi necessário definir a estrutura de rubricas a utilizar. Com isto é possível definir que informação se deverá ter em cada uma das rubricas definidas.

Na figura 18 tem-se a visão da estrutura de rubricas definidas até ao nível 4. Importante dizer -se que existem rubricas que possuem níveis inferiores de informação.

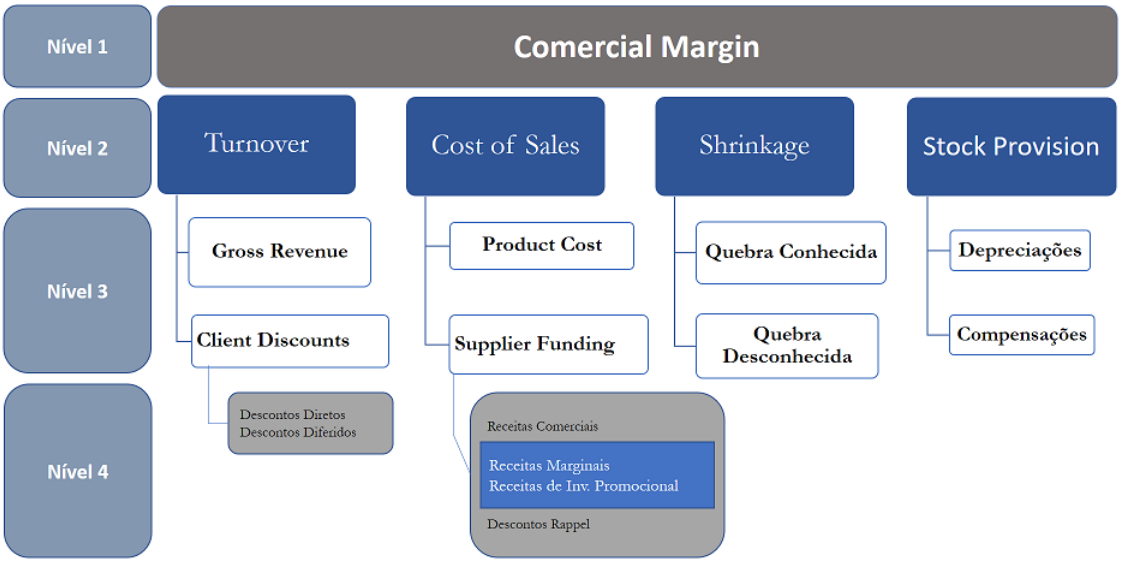


Figura 18 - Árvore de Rubricas.

O primeiro nível identificado é o *Comercial Margin* que abraça todas as rubricas de nível mais baixo. No segundo nível encontram-se a rubrica de *Cost of sales*, *Turnover*, *Shrinkage* e *Stock Provision*. Todas a restantes rubricas identificadas são de nível três, sendo que a rubrica de *Client Discount* e *Supplier Funding* possuem nível quatro identificado na figura 18.

4.4. MODELO MULTIDIMENSIONAL

De forma a idealizar um modelo multidimensional, foi necessário perceber para cada rubrica o processo/área de negócio em questão, as dimensões, os factos e sua respetiva granularidade seguindo as quatro etapas indicadas por Kimball e Ross (2013).

As dimensões identificadas para elaborar e idealizar este modelo foram: (1) dimensão de tempo (dim_time), (2) dimensão de produto (dim_product), (3) dimensão de localização (dim_location), (4) dimensão de conversão (dim_currency), (5) dimensão de sistema (gbl_full_system) e (6) dimensão de rubricas (dim_pnl_account). Todas estas tabelas já existiam, tendo sido apenas necessário a criação da tabela de rubricas.

Importante frisar, que estrutura de informação já existente apresenta a informação com uma granularidade temporal mensal enquanto a nova estrutura de informação terá uma granularidade temporal diária. Com esta definição, os factos neste novo modelo, terão a granularidade de dia, loja, produto e rubrica de nível mais baixo.

A tabela de factos foi designada por Pnla_analysis_d e é comum a todas as rubricas. Esta tabela de factos é particionada de forma diária tendo um subparticionamento pela rubrica de nível dois a que cada rubrica de nível mais baixo está associada. A tabela 4 ilustra a estrutura que a tabela Pnla_analysis_d tem:

Tabela 4 - Estrutura da tabela de factos Pnla_Analysis_D.

PNLA_ANALYSIS_D	
Atributo	**Tipo de Dados**
Time_key	Decimal (8,0)
Product_key	Varchar (128)
Sku	Varchar (128)
Sku_aggr_key	Decimal (38,0)
Location_key	Varchar (128)
Location_cd	Varchar (128)
Loc_aggr_key	Decimal (38,0)
Account_key	Varchar (128)
Currency_key	Varchar (128)
System_key	Decimal (38,0)
Amount	Decimal (20,4)
Amount_eur	Decimal (20,4)
Create_date	Timestamp
Last_updt_date	Timestamp
Partição	
Day_key	Decimal (8,0)
Subpartição	
Account_Domain_key	Varchar (128)

Com isto, resultou o modelo multidimensional em estrela onde existe uma única tabela de factos com as diferentes tabelas. A figura 19 mostra o modelo multidimensional final.

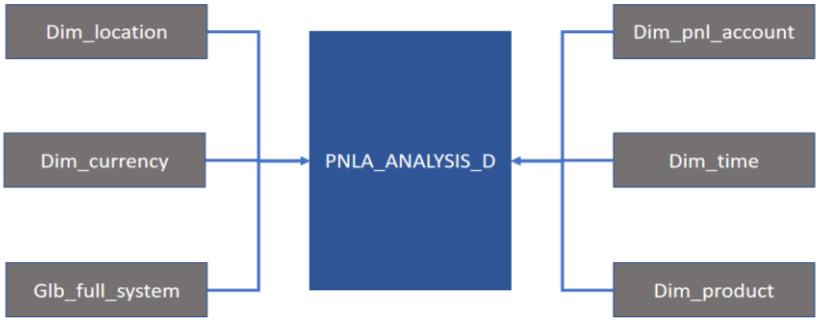


Figura 19 - Modelo Multidimensional.

4.5. ESPECIFICAÇÃO FUNCIONAL

Neste novo modelo, as fórmulas de rateio quer das receitas de investimento promocional, quer das receitas marginais foram alteradas de forma a responder melhor às necessidades das áreas comerciais. Posto isto, é intuito implementar um conjunto de processos para distribuição das mesmas de forma a garantir os seguintes requisitos que estão evidentes na tabela 5.

Tabela 5 - Pressupostos Funcionais de rateio.

Receitas de Investimento Promocional	Receitas Marginais
Reabrir Semanas (Promocionais).	Ratear o valor total da receita no dia de negociação.
Esperar pelo final da promoção para distribuição pelo período da mesma.	Relação fornecedor artigo com base em compras.
Distribuição promocional com base em artigos e lojas associados a promoção.	Rateio por tipo de receita.
Relação fornecedor artigo com base em compras.	
Rateio por tipo de receita.	
Quando, a data negociação – a data fim promoção > 7 dias ou a receita não tem um código promocional valido associado efetuar rateio igual a receitas marginais.	

As receitas de investimento promocional e marginais são identificadas por um código que determina qual é o seu tipo. De frisar, que mesmo sendo uma receita de investimento promocional ou marginal pode ter diferentes tipos. A figura 20 demonstra os diferentes tipos de receitas de investimento promocional e marginal.

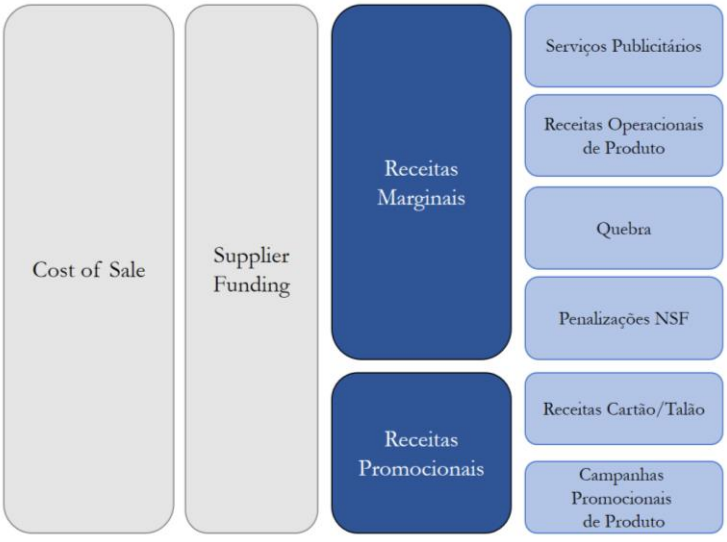


Figura 20 - Diferentes tipos de Receitas Marginais e Promocionais. Elaboração própria.

Face a esses tipos pode-se ter três tipo distribuição:

- **Distribuição Período Promocional** → Distribuição efetuada com base na duração de uma campanha promocional desde que esta não exceda os 45 dias, quando excede são contabilizados os últimos 45 dias. **Elegíveis:** Receitas do tipo promocional.
- **Distribuição na Data de Registo** → Distribuição efetuada na data de registo da receita, com base nos últimos 30 dias de vendas\descontos a contar desde a data de registo. **Elegíveis:** Receitas do tipo marginal ou alguns casos específicos de receitas promocionais.
- **Distribuição equitativa** → Distribuição em que não é possível aplicar uma base de venda/desconto. Nestas situações a distribuição do valor de receita é feita de forma igual por todos os artigos/lojas elegíveis. **Elegíveis**: Receitas do tipo marginal ou promocional, para as quais não existam vendas ou descontos para a receita em questão.

Uma receita pode ser registada a qualquer nível de estrutura mercadológica[2], isto é, um produto está a associado a uma subcategoria de produtos que por sua vez está associada a uma categoria de produtos.

Posto isto, uma receita pode ser registada a nível desta estrutura, no entanto a granularidade definida para a tabela de factos é ao produto. Face a este requisito, uma receita registada ao nível que não ao produto será distribuída apenas por artigos comprados ao fornecedor a que a receita vem registada.

É importante referir que as receitas de investimento promocional para determinação dos produtos que receberam receita têm que ter sempre em consideração os produtos que se encontram na promoção.

[2] Estrutura mercadológica – consiste na estrutura hierárquica dos produtos. Os níveis desta hierarquia do maior para o mais baixo são: (1) direção comercial, (2) unidade de negócio, (3) categoria, (4) subcategoria, (5) unidade base e (6) sku;

Para efetuar a distribuição das receitas tem-se um conjunto bem definidos de regras. As regras de distribuição de receitas marginais são:

- Receita deve ficar totalmente alocada na data de negociação, distribuída ao artigo de acordo com o peso das vendas ou descontos dos últimos 30 dias (data de negociação);
- Apenas para determinadas localizações;
- Não há distribuição semanal, o valor deve ser colocado no próprio dia de negociação, distribuição com base em venda ou descontos;
- Quando a receita é ao artigo mesmo que não exista venda deve ficar no próprio artigo;
- Quando a receita é ao artigo não são validados os artigos associados ao fornecedor, no entanto são validadas as taxas de participação do fornecedor no artigo, quando não exista a taxa é de 100%;
- Quando não existem vendas\descontos para os artigos da lista de compra do fornecedor, o valor é distribuído pelos artigos da lista de forma equitativa.
- Quando não existem artigos na lista de compra do fornecedor, será validado se existem vendas para os artigos em que o fornecedor da receita é o prioritário, caso existam será efetuada distribuição com base em vendas.

Por sua vez, as receitas de investimento promocional, apesar de algumas especificidades a ter em conta, seguem em regra geral de distribuição efetuada pelo processo marginal. As regras especificas para as receitas promocionais são:

- Receita deve ficar totalmente alocada às datas de promoção, independentemente do fecho de semana ou mês, de acordo com o peso das vendas no período da campanha.
- Quando a duração da promoção é superior a 45 dias o valor é distribuído pelos últimos 45 dias anteriores a data de fecho da promoção.
- Apenas quando a data de processo ultrapassar a data fim de promoção é que será distribuído o valor total da receita pelas datas da promoção.
- Distribuição só deve contabilizar artigos e lojas associadas à promoção.

4.6. ESPECIFICAÇÃO TÉCNICA

A nível técnico todas as implementações foram realizadas através da utilização de *HiveQL, Spark* já mencionadas e descritas anteriormente e de um IDE da IBM. Essa interface designa-se por *InfoSphere Datastage*. Este software fornece uma estrutura gráfica para desenvolver processos de manipulação de dados desde sistemas de origem até os sistemas de destino. Suporta os padrões de extração, transformação e carregamento (ETL) e de extração, carregamento e transformação (ELT). O *InfoSphere Datastage* usa processamento paralelo, e é escalável. Possui também a capacidade de integrar dados heterogêneos, incluindo dados grandes em repouso (baseados no *Hadoop*) ou dados grandes em movimento (baseados em fluxo), podendo aplicar regras de negócios. O *InfoSphere Datastage* é bastante intuitivo de utilizar e implementar possibilitando integração de dados em tempo real.

No *InfoSphere Datastage*, vulgarmente designado unicamente por *Datastage*, tem-se dois tipos de *jobs*: (1) *parallel job* e (2) *sequence job*. De forma simplista, o *sequence job* é um orquestrador de outros *jobs*, ou seja, não permite realizar grande lógica, mas sim definir e invocar outros *sequences* e/ou *parallel jobs*. Por sua vez, um *parallel job*, é o local onde se efetua toda a lógica de transformação e agregação da informação.

De modo a sintetizar a utilização do *datastage*, refere-se que o mesmo permite uma organização por pastas. Geralmente estas pasta são organizadas pelas ações de desenvolvimento. Assim, este processo *datastage*, tal como os seguintes, tem cinco conjuntos de ações. A primeira é a recolha de informação da *framework*; a segunda é a criação de tabelas num *working schema* com a informação estritamente necessária. Após este processo, surge o *job datastage,* que irá realizar as transformações necessárias, e inserir a informação numa tabela do *working schema* muito semelhante à final. As duas últimas etapas são a inserção dos dados na tabela final e, por fim, a escrita na *framework* do que foi processado, para que os processos que dependem deste possam saber o que foi ou não processado.

O termo *framework* referido no parágrafo, é muito mais do que o que citado no mesmo, pois consiste numa metodologia de desenvolvimento e implementação definida pela própria BIT.

De um modo geral, todo processo desenvolvido para o Hadoop segue um padrão específico e faz uso de uma estrutura com APIs para registrar a atividade e gerar análises em

cada execução do processo. Resumidamente, é uma estrutura de processos e de como os dados são trabalhados e guardados. Essa estrutura pode ser vista na imagem da figura 21.

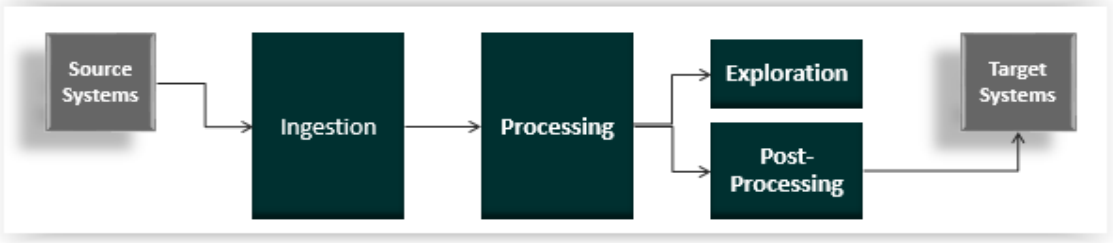


Figura 21 -Framework de desenvolvimento BIT (*Business Information Tecnology*).

Referir que a utilização desta *framework* de desenvolvimento, traduz-se, além de uma melhor e maior organização, em outros benefícios tais como, criar dependências entre processos, saber que dias e que lojas foram processadas pelos processos precedentes de foram a evitar processamentos desnecessários, quantos e quais processos falharam, quantos e quais processos cumprem ou falham o tempo previsto de execução, entre outros.

Após a perceção das tecnologias a utilizar, e de todas as necessidades funcionais, segue-se então a especificação de tarefas a executar e o algoritmo a implementar. Na figura 22 pode-se verificar a arquitetura montada para alimentar a rubrica de receitas marginais e receitas investimento promocional como um todo, tendo perceção dos diferentes processos a construir, tabelas a criar e tabelas já existentes e que estarão ligadas a estes novos processos.

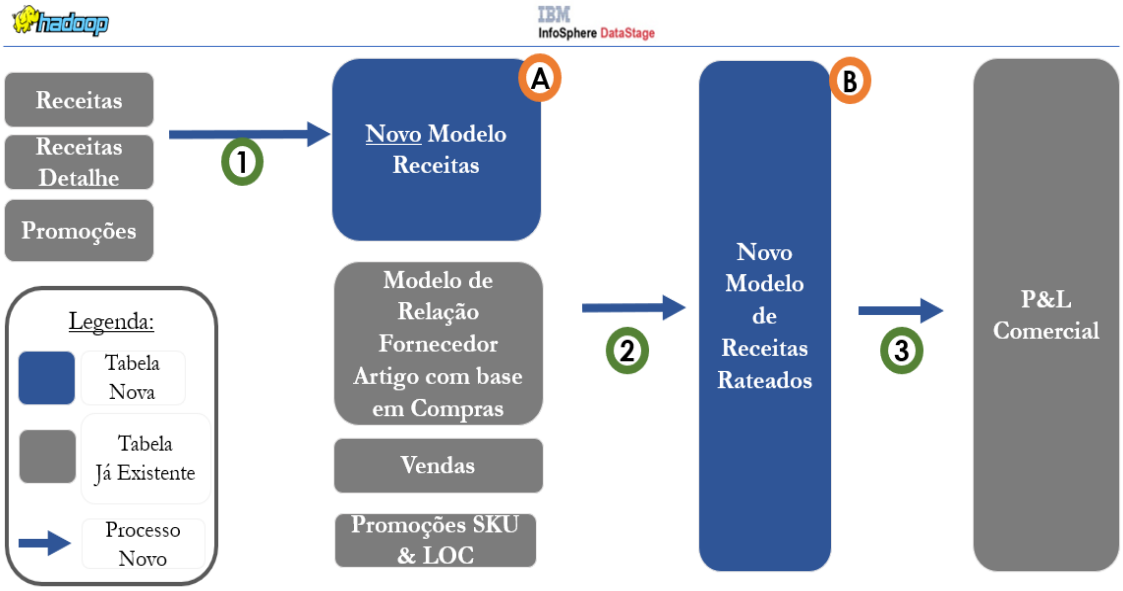


Figura 22 - Arquitetura de processos das rubricas de receitas marginais e promocionais.

De uma forma clara pode-se verificar que foi necessário criar um total de três novos processos, identificados na figura 22 pelos números (1), (2) e (3). Foi necessário também criar várias tabelas identificadas na imagem anterior pelas letras (A) e (B).

A criação das tabelas e consequentemente as sua estruturas foram sempre pensadas na ótica de serem capazes de guardar informação, o mais genérico e reutilizável por outros processos, mas sempre com o foco em responder ao intuito deste desenvolvimento. É de frisar que a tabela identificada com a letra (B) se trata de um conjunto de tabelas.

Na tabela 6 encontra-se o nome atribuído a cada tabela e uma breve descrição das mesmas.

Tabela 6 - Tabela com designação e breve descrição das tabelas a criar.

Identificação Figura 22	Designação	Descrição
A	Revf_com_deal_dot_neg_prom	Nova tabela de Receitas, junta as tabelas anteriores com a tabela já existente de promoções.
B	Reva_marg_d	Tabela final de receitas marginais após processo de rateio.
	Reva_prom_d	Tabela final de receitas promocionais após processo de rateio.

A nível de processos foi necessário a criação de raiz dos processos (1), (2) e (3), uma vez que são processos que não existiam. De notar que os processos identificados com os números (2) e (3), não são um único processo mas englobam um conjunto de processos a implementar. Na tabela 7, tem-se o nome atribuído a cada tabela e uma breve descrição.

Tabela 7 - Tabela com designação e breve descrição dos processos a criar.

Identificação Figura 22	Designação	Descrição
1	Modelo Receitas	Novo modelo base que agrega informação pela data de negociação (Data de Registo), junta também informação de cabeçalho com detalhe (Informação Mercadológica) e data início e fim de promoção para receitas do tipo promocional.
2	Algoritmos de Distribuição	Processo responsável pela distribuição das receitas marginais e receitas investimento promocional.
3	Integração P&L Comercial	Processo de disponibilização e agregação de tipos de receita para carregamento da rubrica de receitas marginais e receitas investimento promocional.

De forma a especificar os algoritmos foram elaborados fluxogramas para melhor entendimento dos rateios a efetuar. Nos anexos II, III e V encontra-se a explicação dos pontos de decisão e das ações a realizar dos dois algoritmos, fluxogramas do algoritmo de distribuição de receitas marginais e de receitas de investimento promocional, respetivamente. Nos anexos IV e VI tem-se a explicação dos diferentes outputs dos algoritmos.

5. IMPLEMENTAÇÃO

Uma implementação deste tipo, por norma segue sempre as boas práticas estipuladas por cada organização. A BIT não é exceção e por isso existem um conjunto de ações que são comuns e quase transversais a todos os desenvolvimentos.

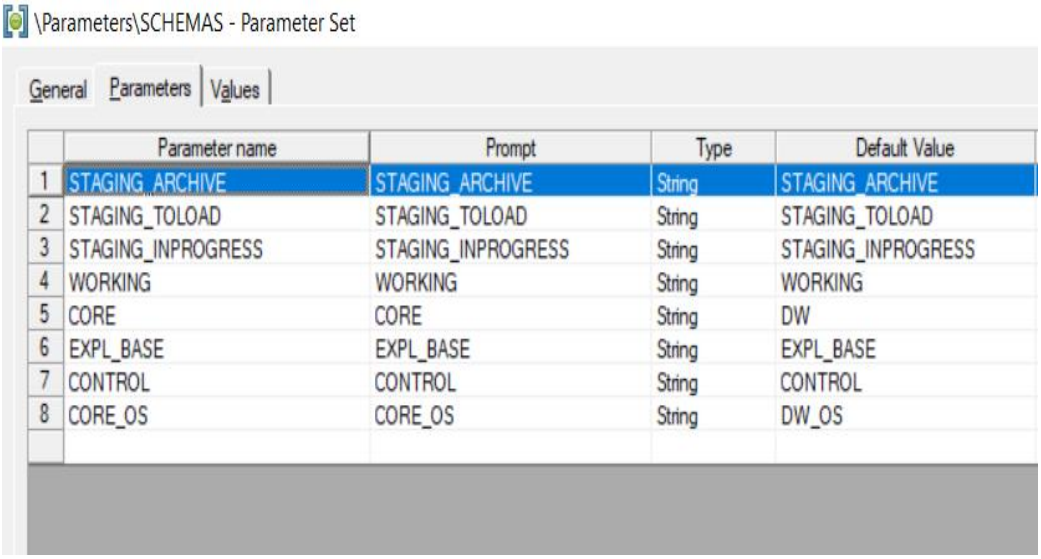
Um desses casos é a parametrização do processo numa *framework* instituída na BIT. Esta *framework* permite criar dependências entre outros processos, saber a última execução com sucesso e até mesmo que dias processou. Permite com isto uma melhor eficiência pois caso o processo que é dependência de outro não tenha executado o nosso nada processará. Para usufruir destas funcionalidades deve-se parametrizar o processo através de um comando e criar o ficheiro de dependências. Esse mesmo ficheiro tem que ser inserido num diretório da máquina de *Hadoop*. Na tabela 8 pode ser visto o comando genérico de parametrização na *framework*.

Tabela 8 - Exemplo genérico de parametrização da *framework*.

Comando genérico de parametrização da *framework*
ds*job* -run -wait -mode RESET *FRAMEWORK* js_post_*job*.<nome_processo>_<tipo_processo>;
ds*job* -run -*job*status -warn 0 \
-param *FRAMEWORK*="<ambiente>" \
-param param_process="<nome_processo>" \
-param param_stage="<tipo_processo>" \
-param param_domain="<dominio>" \
-param param_process_type="<tipo_tabela>" \

```
-param param_deadline="" \
-param param_sub_domain="" \
-param param_status="ACTIVE" \
-param sourceFilePath="<path_diretório>" \
-param sourceFilenameSources="<nome_do_ficheiro_de_fontes>" \
-param sourceFilenameDependencies="<nome_do_ficheiro_de_dependencias>" \
FRAMEWORK js_post_job.<nome_processo>_<tipo_processo>
```

Outra ação que é comum é a reutilização de *parameters sets*. Este *parameters sets* são comuns a todos os projetos, pois trata-se de um conjunto de informação comum. Esta informação vai desde os diretórios, *schemas*, ligação, paralelismo, entre outros. No entanto, é comum que cada processo possua um *parameter set* específico como poderá ser visto em seguida. Na imagem da figura 23, está espelhado um exemplo de um *parameter set*.



	Parameter name	Prompt	Type	Default Value
1	STAGING_ARCHIVE	STAGING_ARCHIVE	String	STAGING_ARCHIVE
2	STAGING_TOLOAD	STAGING_TOLOAD	String	STAGING_TOLOAD
3	STAGING_INPROGRESS	STAGING_INPROGRESS	String	STAGING_INPROGRESS
4	WORKING	WORKING	String	WORKING
5	CORE	CORE	String	DW
6	EXPL_BASE	EXPL_BASE	String	EXPL_BASE
7	CONTROL	CONTROL	String	CONTROL
8	CORE_OS	CORE_OS	String	DW_OS

Figura 23 - Exemplo de um *parameter set* geral aos diferentes projetos.

5.1. PROCESSO DE CRIAÇÃO DO NOVO MODELO DE RECEITAS

Este processo consiste em relacionar as tabelas referidas no processo de sincronização com a informação de promoção. Uma vez que não existia nenhum processo que realizasse esta ação foi necessário desenvolver da raiz este processo. Para tal desenvolvimento, tal como já citado, foi utilizado o *Datastage*. Este processo ficou designado por revf_com_deal_dot_neg_prom_proc.

Este desenvolvimento passou por três grandes etapas que são:

- Parametrização da *framework*;
- Criação da tabela final;
- Desenvolvimento em *Datastage*;

O registo na *framework* deste processo seguiu o processo genérico apenas preenchendo o comando genérico já mostrado anteriormente. O comando de parametrização para este processo é então o que se encontra na Tabela 9.

Tabela 9 - Comando de parametrização do processo revf_com_deal_dot_neg_prom_proc.

Comando de parametrização do processo revf_com_deal_dot_neg_prom_proc
ds*job* -run -wait -mode RESET *FRAMEWORK* js_post_*job*.revf_com_deal_dot_neg_prom_proc;
ds*job* -run -*job*status -warn 0 \
-param *FRAMEWORK*="PRD" \
-param param_process="revf_com_deal_dot_neg_prom" \
-param param_stage="proc" \
-param param_domain="profit_loss" \
-param param_process_type="profit_loss" \
-param param_deadline="" \
-param param_sub_domain="" \
-param param_status="ACTIVE" \
-param sourceFilePath="/opt/IBM/Projects/ds_files/profit_loss/*framework*/" \
-param sourceFilenameSources="sources_*Hadoop*" \
-param sourceFilenameDependencies="dependencies_revf_com_deal_dot_neg_prom" \
FRAMEWORK js_post_*job*.revf_com_deal_dot_neg_prom_proc

É importante frisar que o nome do ficheiro indicado no *script* de parametrização indica o nome dos processos de sincronização anteriores.

A tabela final que este processo passou a alimentar foi designada por revf_com_deal_dot_neg_prom e na sua estrutura possui atributos que também estão presentes nas tabelas intervenientes neste processo. As tabelas que contribuem com informação neste processo são:

- Revf_com_deal_dot_neg;
- Revf_com_deal_dot_det;
- Dim_prom_promotion;

No anexo VII pode-se constatar a estrutura da tabela revf_com_deal_dot_neg_prom bem como o *script* da sua criação. Importante referir que alguns atributos são para facilitar os processos que dependeram deste. Esses atributos possuíram desde o tipo de rateio a efetuar, a diferença entre a data de negociação e a de fim de promoção, entre outros.

Após a realização da primeira e segunda etapa segue -se então o desenvolvimento em *datastage*. A figura 24 apresenta a estrutura de pastas definidas para este processo.

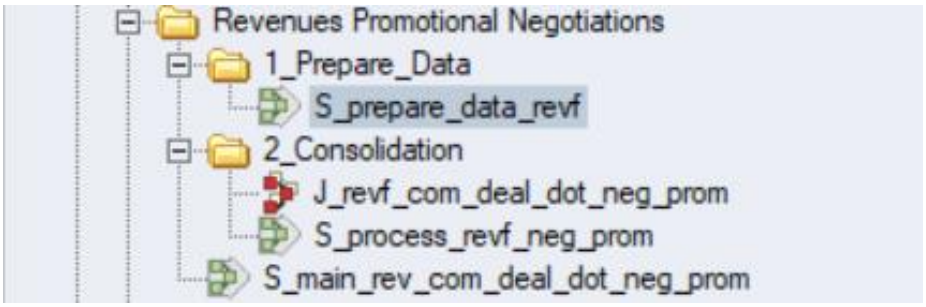


Figura 24 - Estrutura de pasta no *Datastage* do processo revf_com_deal_dot_neg_prom.

Seguindo a ordem de execução, mas não a ordem de desenvolvimento, o primeiro *job* é o S_main_com_deal_dot_neg_prom. Este *job* consiste num *job* quase genérico uma vez que, com mais ou menos lógica, irá fazer sempre as mesmas ações. Este verificará os dias que as dependências processaram e injetá-los no *job* S_process_revf_com_neg_prom. No final da execução deste último, irá escrever na *framework* os dias que acabou de processar.

O *job* S_process_revf_com_neg_prom é outro orquestrador onde aqui é invocado primeiramente o *sequence job* S_prepare_data_revf e depois o *parallel job* J_revf_com_deal_dot_neg_prom.

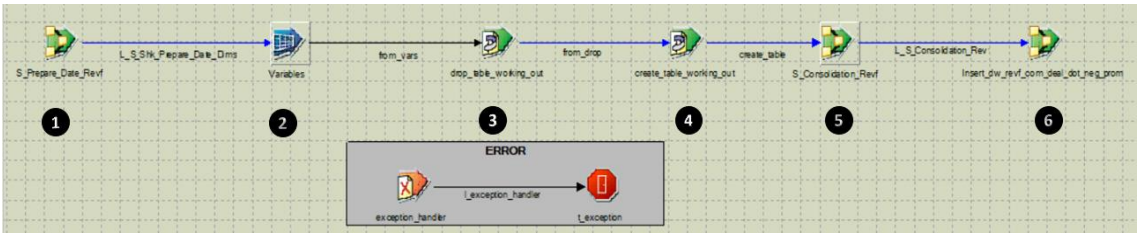


Figura 25 - *Sequence job* - S_process_revf_com_neg_prom

Na figura 25 vê-se a estrutura montada no *datastage*. Cada número na imagem corresponde a um componente que executa as seguintes funções:

1. [S_prepare_Data_Revf] – Invoca o *sequence job* S_prepare_data_revf, passando-lhe os dias a processar.
2. [Variables] – Componente onde se coloca os valores para variáveis que se pretenda utilizar neste *sequence job*. Neste caso, possui o *script* de *drop* e criação da tabela de *working* onde o *parallel job* insere os dados finais.
3. [Drop_table_working_out] – Efetua o comando de *drop* da tabela de *working*. Efetua-se sempre primeiro o comando de *drop* de forma a salvaguardar que a tabela não possui qualquer informação de uma inserção anterior.
4. [Create_table_working_out] - Efetua o comando de create da tabela de working.
5. [S_consolidation_revf] – Invoca o *parallel job* J_revf_com_deal_dot_neg_prom.
6. [Insert_dw_revf_com_deal_dot_neg_prom] – Efetua o *insert* da informação que se encontra na tabela de *working* na tabela de DW. O *script* de insert encontra-se no próprio componente.

Seguindo então a ordem de execução o próximo *job*, é outro *sequence job*, mas com características diferentes. O S_prepare_data_revf possui um conjunto de componentes que invocam um *job* genérico que apaga e cria as tabelas no *working schema*. Esse *job* genérico recebe deste *job*, especificamente de cada componente, o nome a atribuir à tabela a criar, bem como o comando que seleciona a informação que queremos que cada tabela de *working* tenha. Este comando não é nada mais nada menos que um *select* aos atributos que se pretende ter nas tabelas criadas. É de frisar que se pode colocar alguma lógica como conversão de tipo de dados, *joins* entre tabelas, limitação temporal, entre outros.

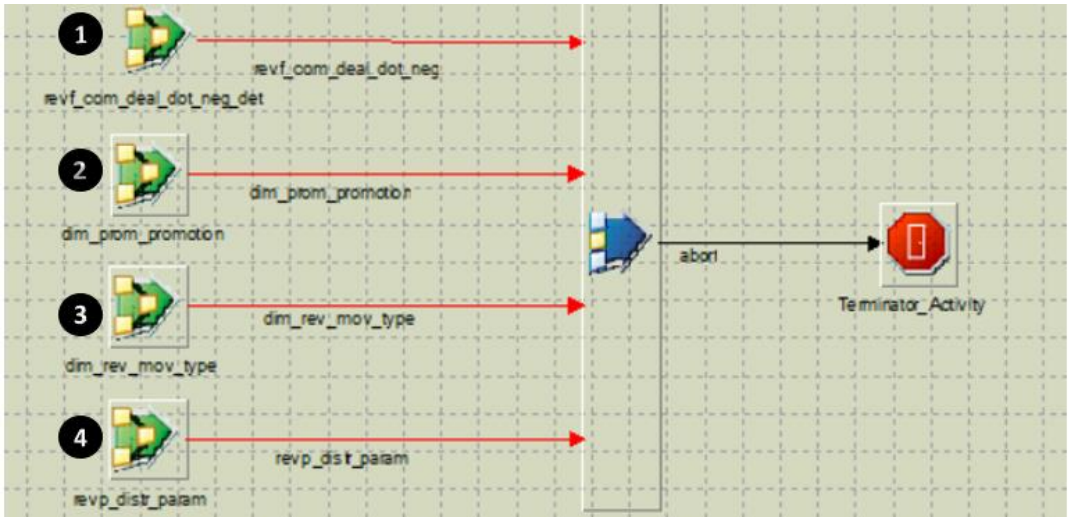


Figura 26 - *Sequence job* - S_prepare_data_revf

A figura 26 ilustra a estrutura montada no *datastage*. Cada número na imagem correspondes a um componente que executa a seguinte função:

1. [Revf_com_deal_dot_neg_det] – Criação de uma tabela de working com informação da tabela revf_com_deal_dot_neg e revf_com_deal_dot_det.
2. [Dim_prom_promotion] – Criação de uma tabela de *working* com informação da tabela dim_prom_promotion.
3. [Dim_rev_mov_type] – Criação de uma tabela de *working* com informação da tabela dim_rev_mov_type.
4. [Rep_distr_param] - Criação de uma tabela de *working* com informação da tabela revp_distr_param.

O último *job* invocado é o J_revf_com_deal_dot_neg_prom, sendo que este é o único *parallel job* deste processo. Na figura 27 vê-se a estrutura implementada no *datastage*.

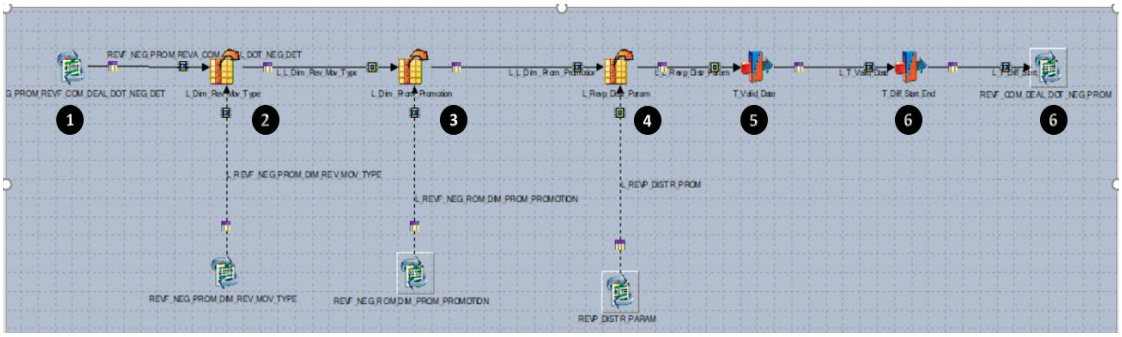


Figura 27 - *Parallel job* - J_revf_com_deal_dot_neg_prom.

O componente identificado na figura 27, com o número 1, trata-se de um FileConnector. Este componente acede ao HDFS correspondente da tabela de *working* com informação da tabela revf_com_deal_dot_neg e revf_com_deal_dot_det.

Os componentes identificados com os números 2, 3 e 4 tratam-se de *lookups*. Estes componentes realizam a ação de efetuar *left* ou *inner join* entre tabelas. Neste caso pode-se ver que no número 2 se efetua o *join* da informação da tabela de *working* com informação das tabelas revf_com_deal_dot_neg e revf_com_deal_dot_det com a tabela dim_prom_promotion. O componente 3 efetua a mesma ação, mas com a informação resultante da junção realizada pelo componente 2 e a tabela outra das tabelas criadas em *working*.

Os componentes 5 e 6 são *transforms*. Estes componentes permitem implementar funções condicionais, de conversão, entre outras. No componente 5 está a fazer uma validação ao ano que se encontra na data de fim de promoção. Por sua vez, o componente 6 calcula por exemplo a diferença entre a data de negociação e a data de fim de promoção devolvendo o resultado em dias. Para este cálculo recorreu-se a uma função do *datastage* DaysSinceFromDate2(DateA,DateB).

Por sua vez o componente 6 trata-se novamente de um FileConnetor mas este em vez de ser de leitura trata-se de um com funcionalidades de escrita escrevendo assim o resultado destas ações na tabela de *working* criado no *sequence job* S_process_revf_com_neg_prom. Após a execução deste *job*, segue a inserção na tabela final no *schema* de DW que se encontra no *job* S_process_revf_com_neg_prom. O *script* de insert utilizado encontra-se na tabela 10.

Tabela 10 - Insert final do processo revf_com_deal_dot_neg_prom.

Insert final do processo revf_com_deal_dot_neg_prom

set hive.exec.parallel=true;
set hive.exec.dynamic.partition=true;
set hive.exec.dynamic.partition.mode=nonstrict;
INSERT OVERWRITE TABLE ":*SCHEMAS*.CORE:".":"revf_com_deal_dot_neg_prom
PARTITION(month_key)
SELECT a.time_key, a.location_key, a.supplier_key, a.rev_type_key, a.promotion_cd,
a.prod_hierarchy_key, a.contract_no, a.sequence_no, a.negotiation_date, a.int_dot_val, a.int_curr_key,
a.sup_dot_val, a.sup_curr_key, a.biz_unit_cd, a.cat_cd, a.subcat_cd, a.brand_cd, a.sku,
a.prom_start_date, a.prom_end_date, a.dif_neg_prom, a.calc_type, a.prom_valid, a.system_key,
a.create_date, a.last_updt_date, SUBSTR(a.time_key, 0, 6) AS month_key
FROM ":*SCHEMAS*.WORKING:".":"revf_com_Deal_dot_neg_prom a
WHERE a.time_key IN (":TIME_KEY_LIST:")
UNION ALL
SELECT b.time_key, b.location_key, b.supplier_key, b.rev_type_key, b.promotion_cd,
b.prod_hierarchy_key, b.contract_no, b.sequence_no, b.negotiation_date, b.int_dot_val,
b.int_curr_key, b.sup_dot_val, b.sup_curr_key, b.biz_unit_cd, b.cat_cd, b.subcat_cd, b.brand_cd,
b.sku, b.prom_start_date, b.prom_end_date, b.dif_neg_prom, b.calc_type, b.prom_valid, b.system_key,
b.create_date, b.last_updt_date, b.month_key
FROM ":*SCHEMAS*.CORE:".":"revf_com_Deal_dot_neg_prom b
WHERE b.time_key NOT IN (":TIME_KEY_LIST:")
AND b.month_key IN (":MONTH_KEY_LIST:");

Após o *insert*, o fluxo regressa ao primeiro *job*, o S_main_com_deal_dot_neg_prom, que como última ação deste processo escreverá os dias que este processo utilizou na execução.

5.2. PROCESSO DE DISTRIBUIÇÃO DE RECEITAS MARGINAIS

Este processo consiste em efetuar a distribuição de receitas marginais. Uma vez que não existia nenhum processo que realizasse esta ação foi necessário desenvolver da raiz. Para tal desenvolvimento, tal como já citado, foi utilizado o *Datastage*. Este processo ficou designado por reva_marg_d_proc. À semelhança de outros processos, este desenvolvimento passou por três grandes etapas que são: a parametrização da *framework*; criação da tabela final; e, por último, o desenvolvimento em *Datastage*.

O registo na *framework* deste processo seguiu o processo genérico apenas preenchendo o comando genérico já mostrado anteriormente. O comando de parametrização para este processo é então o que se encontra na tabela 11.

Tabela 11 - Comando de parametrização do processo reva_marg_d_proc.

Comando de parametrização do processo reva_marg_d_proc
/opt/IBM/InformationServer/Server/DSEngine/bin/ds*job* -run -wait -mode RESET *FRAMEWORK* js_post_*job*.reva_marg_d_proc; /opt/IBM/InformationServer/Server/DSEngine/bin/ds*job* -run -*job*status -warn 0 \ -param *FRAMEWORK*="PRD" \ -param param_process="reva_marg_d" \ -param param_stage="proc" \ -param param_domain="profit_loss" \ -param param_process_type="profit_loss" \ -param param_deadline="" \ -param param_sub_domain="" \ -param param_status="ACTIVE" \ -param sourceFilePath="/opt/IBM/Projects/ds_files/profit_loss/*framework*/" \ -param sourceFilenameSources="sources_*Hadoop*" \ -param sourceFilenameDependencies="dependencies_reva_marg_d" \ *FRAMEWORK* js_post_*job*.reva_marg_d_proc

A tabela final que este processo passou a alimentar foi designada por reva_marg_d e procurou-se que a sua estrutura fosse idêntica à estrutura da tabela precedente. Este processo

é mais complexo e, consequentemente, possui muitas tabelas intervenientes no processo. As tabelas intervenientes neste processo são:

- Revf_com_deal_dot_neg_prom;
- Slsa_reporterd_net_sale_d;
- Pnla_all_Discounts;
- Pura_avg_cost_ratio.

É ainda importante frisar, que este processo também utiliza tabelas de dimensão e parametrização.

No anexo VIII pode-se ver a estrutura da tabela reva_marg_d bem como o *script* da sua criação. Após a realização da primeira e segunda etapa segue -se então o desenvolvimento em *datastage*. Na figura 28 pode-se visualizar a estrutura de pastas definidas para este processo.

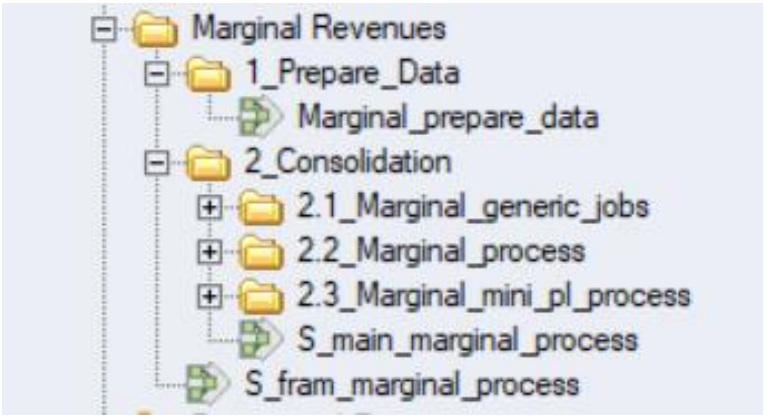


Figura 28 - Estrutura de pasta no *Datastage* do processo reva_marg_d.

A estrutura de definida para este processo, devido à complexidade deste processo, e como se pode verificar na figura 28, é um pouco diferente. De forma a perceber esta estrutura, pode-se afirmar que o desenvolvimento em *datastage* deste processo pode ser divido em 5 grandes etapas:

- Preparação das tabelas a utilizar para o *working schema*;
- Preparação das receitas – explosão das receitas que são cadastradas são a um nível diferente do artigo, baseado nas tabelas de compras;
- Preparação da informação de venda/descontos dos últimos 30 dias;
- Distribuição das receitas;
- Inserção na tabela final reva_marg_d;

No entanto, seguindo a ordem de execução, mas não a ordem de desenvolvimento, o primeiro *job* é o S_fram_marginal_process. Este *job* consiste num *job* quase genérico uma vez que com mais ou menos lógica irá fazer sempre as mesmas ações. Este *job* irá verificar os dias que as dependências processaram invocar e injetar esses mesmo dias no *job* S_main_marginal_process e no final da execução deste último referido escrever na *framework* os dias que acabou de processar.

De forma a perceber o *sequence job* S_main_marginal_process as figuras 29, 30 e 31 demonstram a estrutura do *job*. Importante que cada número nas imagens corresponde a um componente, ou conjunto de componentes que executam a função descrita posteriormente.

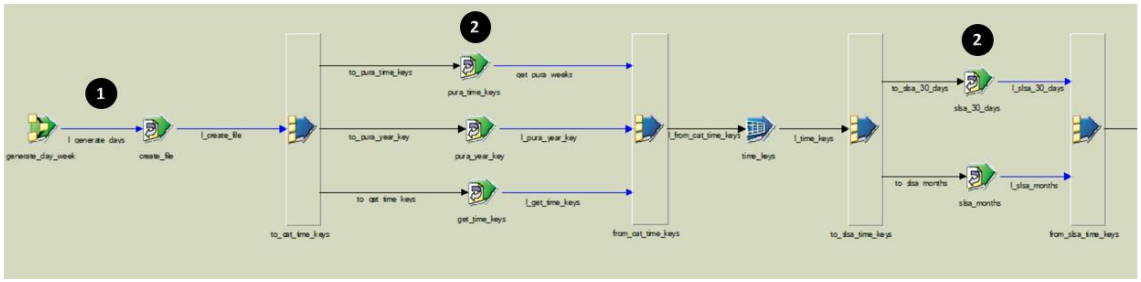


Figura 29 - *Sequence job* - S_main_marginal_process – parte 1/3.

1. Materializa num ficheiro a 2ª feira da semana respetiva ao *time_key* recebido;
2. Calcula os conjuntos de datas para processamento, com base nos *time_keys* recebidos para usar no processo;

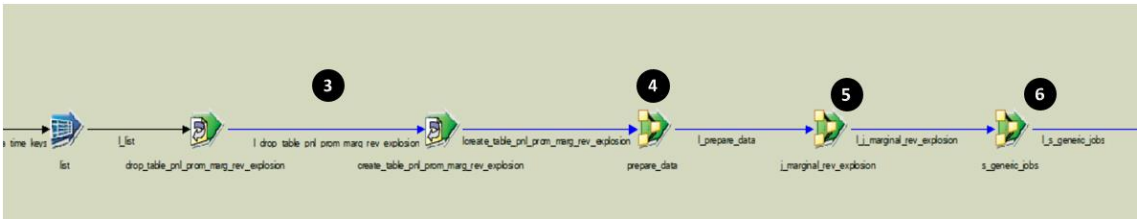


Figura 30 - *Sequence job* - S_main_marginal_process – parte 2/3.

3. Executa as *query*s responsáveis pelo *drop/create* das tabelas *working*
4. S_Promotionl_marginal_prepare_data] Executa o *sequence job* responsável pela preparação de dados para o processo de receitas marginais;
5. [j_marg_prom_explosion] Executa o *parallel job* responsável pela explosão das receitas marginais ao artigo;

6. [S_generic_*job*s] Executa o *sequence job* responsável pelo processamento dos *job*s genéricos para as receitas marginais;

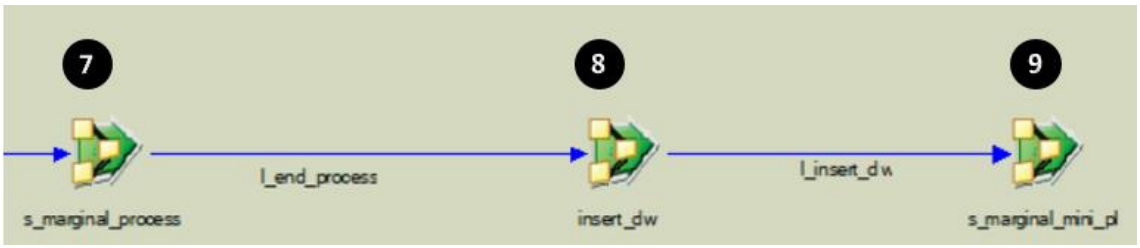


Figura 31 - *Sequence job* - S_main_marginal_process – parte 3/3.

7. [S_marginal_process] Executa o *sequence job* responsável pelo processo de receitas marginais;
8. [Execute_Beeline] Executa a *query* responsável pelo *merge* na tabela final, reva_marg_d;
9. [S_marginal_mini_pl] Invoca o processo que efetua a agregação e respetiva inserção na tabela pnla_analysis_d;

O primeiro *job* a ser invocado pelo *job* descrito anteriormente é o marginal_prepare_data, sendo esta também a primeira etapa identificada. Este *job* possui um conjunto de componentes que invocam um *job* genérico com o intuito de criar tabelas no *working schema* coma informação pretendida e podendo ou não já ter alguma lógica associada. Este comando é um *select* aos atributos que se pretendem nas tabelas criadas. Na figura 32, pode ser visto o *sequence job* referido anteriormente.

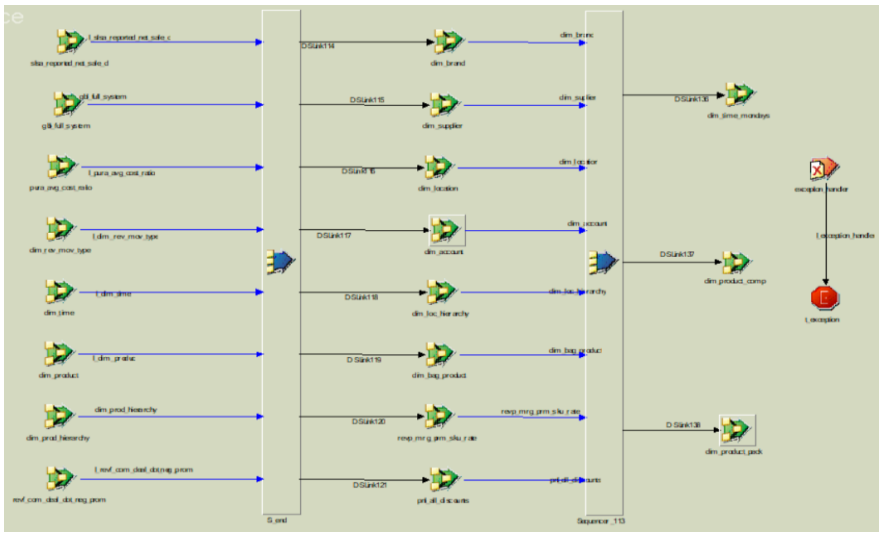


Figura 32 - *Sequence job* - Marginal_prepare_data.

A segunda etapa definida referia-se à explosão das receitas que são cadastradas à estrutura mercadológica diferente do artigo, baseado nas tabelas de compras. Esta ação é efetua no *parallel job* designado por J_marginal_process_rev_explosion. Nas figuras 33 e 34 pode-se verificar o desenvolvimento do *job*.

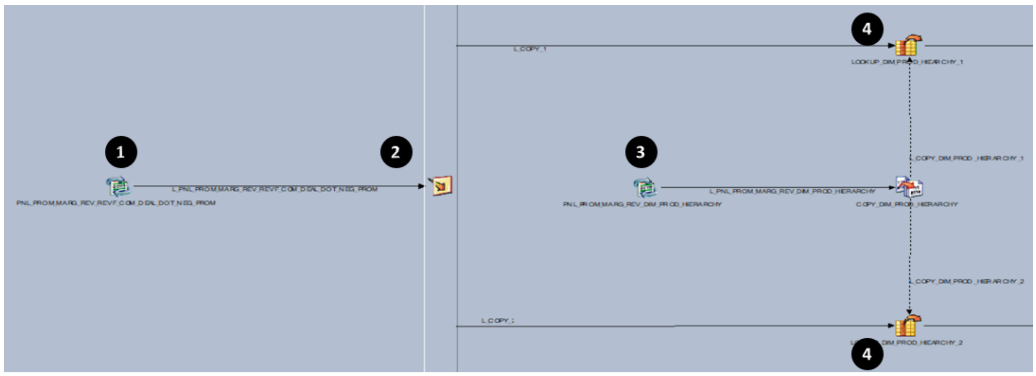


Figura 33 -*Parallel job* - J_marginal_process_rev_explosion - parte 1/2.

1. Carrega os dados da tabela de receitas;
2. Filtra o fluxo com e sem SKU e tipos de receitas;
3. Carrega os dados da tabela dim_prod_hierarchy;

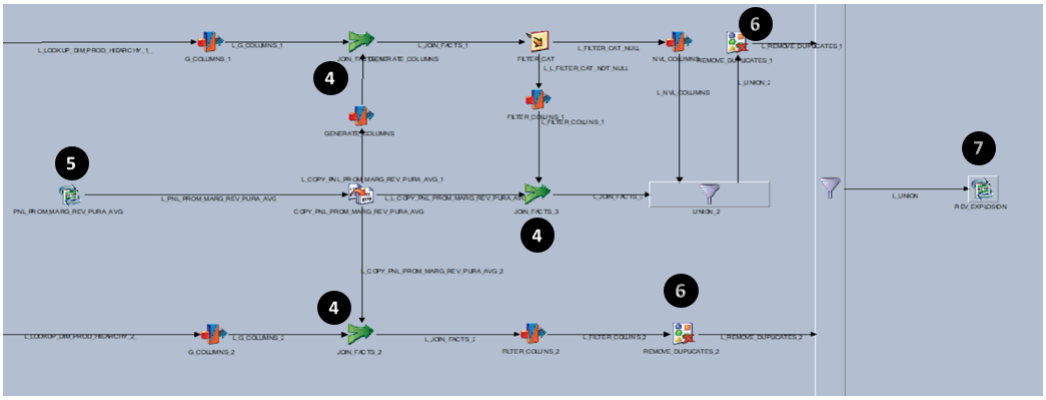


Figura 34 - *Parallel job* - J_marginal_process_rev_explosion - parte 2/2.

4. Lookup às dimensões necessárias ao processo;
5. Carrega os dados da tabela de compras;
6. Remove os duplicados;
7. Escreve os novos dados para a tabela temporária working.pnl_prom_marg_rev_explosion.

De frisar que a nível de estrutura este *job* de desdobramento se encontra juntamente com os *job*s da etapa 3, na estrutura de pasta tal como se pode verificar na figura 35.

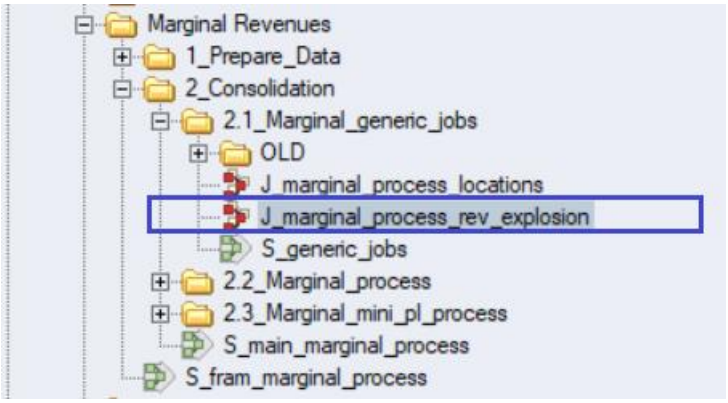


Figura 35 - Identificação dos *job*s da etapa 2 e 3 na estrutura de pastas do processo reva_marg_d.

A terceira etapa identifica é a preparação das vendas/descontos dos últimos 30 dias. *S_generic_jobs* foi o nome atribuído ao *job* com essa função. Este *job* tem como intuito agregar esses 30 dias num único dia, o dia de negociação. A forma mais eficaz foi fazer a agregação via HiveQL. Ou seja, os componentes que se podem ver na figura 36 possuem os *script*s para cada agregação.

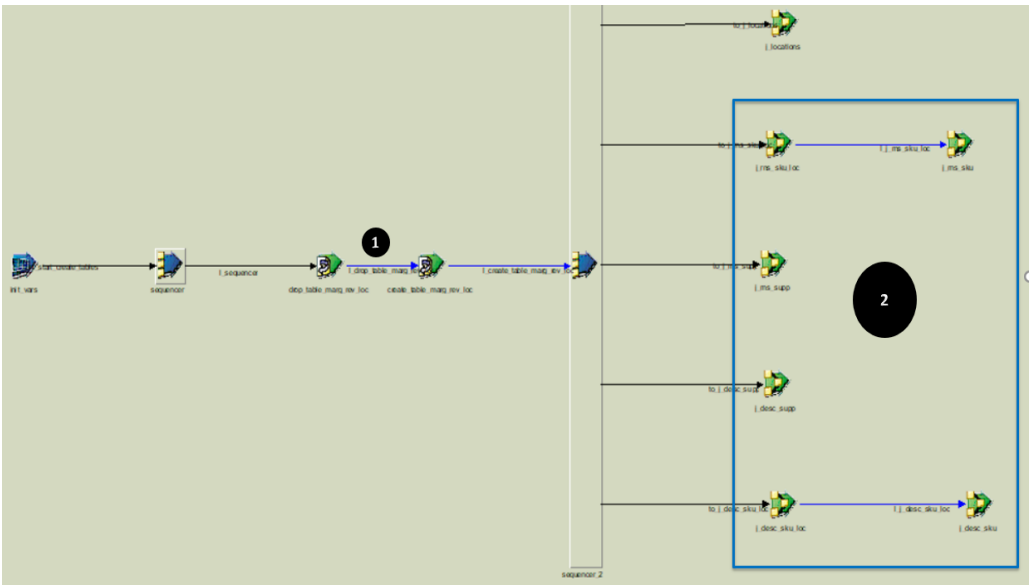


Figura 36 - *Sequence job* – S_generic_*job*s.

Na imagem da figura 36 o conjunto de componentes identificado com o número 1, efetua *drop* e *create* de uma tabela *working*. Por sua vez, os componentes identificados com o número utilizam os tais *script*s para efetuar as agregações da informação a diferentes níveis. Para melhor perceção deste *job* na tabela 12 pode-se verificar um dos *script*s utilizados.

Tabela 12 - *Script* de agregação de informação de vendas - loja x produto.

Agregação de informação de vendas - loja x produto

```
SELECT dt1.time_key,
       rns.location_cd,
       rns.loc_aggr_key,
       rns.sku,
       rns.sku_aggr_key,
       cast(sum(rns.net_sale) AS DOUBLE) AS net_sale,
       cast(sum(rns.net_sale_eur) AS DOUBLE) AS net_sale_eur,
       1 AS occur
FROM ":SCHEMAS.CORE:".dim_time dt1,
    ":SCHEMAS.CORE:".dim_time dt2
INNER JOIN ":SCHEMAS.WORKING:".pnl_marg_rev_slsa_reported_net_sale_d rns ON
dt2.time_key = rns.time_key
WHERE dt1.time_key IN (":TIME_KEYS:")
  AND dt2.seq BETWEEN dt1.seq - 30 AND dt1.seq
GROUP BY dt1.time_key, rns.location_cd, rns.loc_aggr_key, rns.sku, rns.sku_aggr_key
HAVING sum(rns.net_sale) > 0;
```

A quarta etapa é onde se encontra a finalidade deste processo. Para melhorar a forma de desenvolvimento, foi decidido ter *parallels jobs*, um para cada grupo de rubricas identificado. Importante que nas figuras apenas um dos *jobs* será explicado pois os outros possuem algumas diferenças. A estrutura de pastas definidas para esta etapa está clara na figura 37.

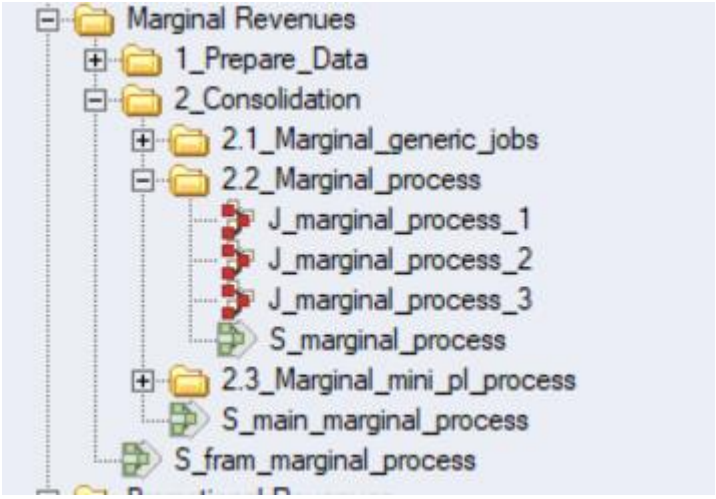


Figura 37 - Identificação dos métodos de rateio na estrutura de pasta do processo reva_marg_d.

As diferenças entre o parralel *job* identificado por j_marginal_process_1 para o j_marginal_process_3 é que o primeiro considera 30 dias resultantes da informação de vendas enquanto o outro considera informação de descontos. Por sua vez o j_marginal_process_2 é identifico ao j_marginal_process_1 pois considera a informação de vendas, mas não ignora a localização que a receita é cadastrada mantendo-a ao contrario do j_marginal_process_1. Importante dizer que é no *sequence job* identificado por S_marginal_process que este parallels *job*s são invocados. É também neste *job* que em todas as execuções se efetua *drop* e *create* das tabelas que estes *job*s alimentam no *working schema*.

Nas Figuras 38, 39 e 40 tem-se então uma explicação da lógica que foi implementada neste três *parallels job*s, mais especificamente no j_marginal_process_1.

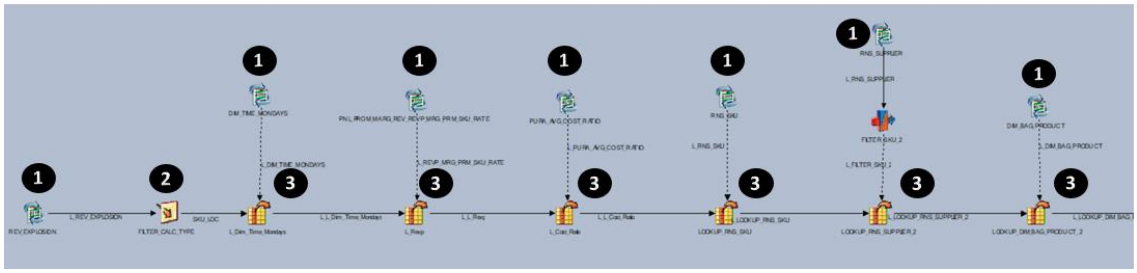


Figura 38 - *Parallel job* - J_marginal_process_1 - parte 1/3.

1. Carrega os dados das tabelas necessárias ao processo;
2. Filtra o método a usar ('M_NH_1','M_S_1');
3. Lookup às dimensões necessárias ao processo;

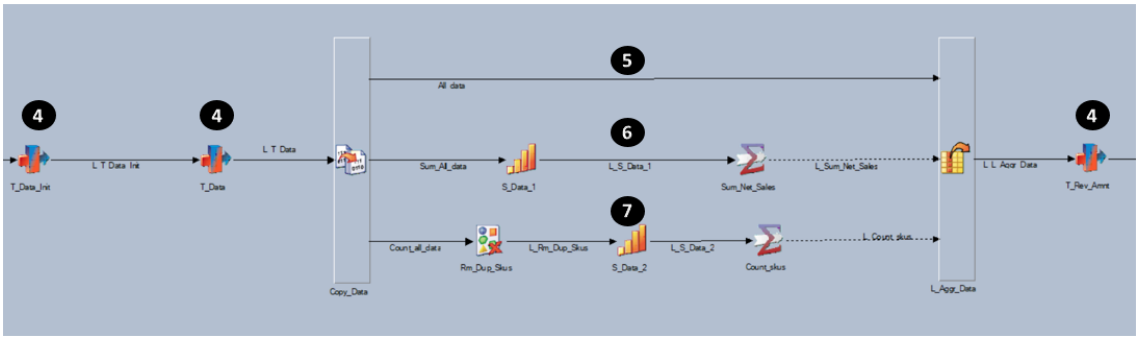


Figura 39 - *Parallel job* - J_marginal_process_1 - parte 2/3.

4. Transformações necessárias ao processo;
5. Envio dos dados sem transformações;
6. Agrega a informação e soma valor líquido da venda;

7. Agrega os dados e conta o número de artigos;

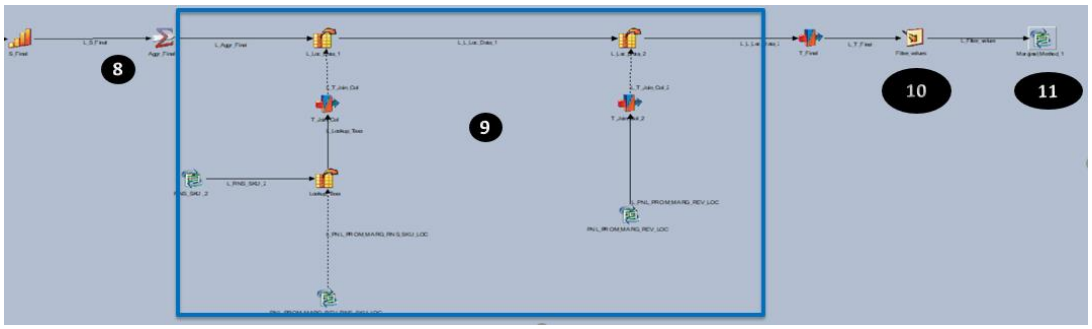


Figura 40 - *Parallel job* - J_marginal_process_1 - parte 3/3.

8. Agrega a informação e soma os valores de receitas;
9. Distribuição à loja.
10. Filtra valores diferentes de 0;
11. Escreve os novos dados para a tabela temporária working.pnl_prom_marg_rev_marginal_method_1.

Após a execução deste *job*, segue a inserção na tabela final no schema de DW que se encontra no *job* S_main_marginal_process. O *script* de insert utilizado encontra-se no anexo IX. Dizer só que antes de este processo fechar a sua execução escrevendo os dias que processou ainda invoca o *job* de inserção na tabela pnla_analysis_d. Este *job* será explicado posteriormente.

5.3. PROCESSO DE DISTRIBUIÇÃO DE RECEITAS DE INVESTIMENTO PROMOCIONAIS

Este processo é muito semelhante ao descrito anteriormente. Consiste em efetuar a distribuição de receitas de investimento promocional. Este processo, à imagem do anterior, também foi desenvolvido de raíz, uma vez que não havia nenhum processo que respondesse às necessidades que este veio responder. Para tal desenvolvimento foi utilizado a mesma ferramenta, juntamente com a utilização de *Spark* em algumas fases do processo. Este processo ficou designado por reva_prom_d_proc. Mais uma vez, alinhado com os anteriores, este desenvolvimento passou pelas mesmas etapas dos outros desenvolvimentos.

O registo na *framework* deste processo seguiu o processo genérico apenas preenchendo o comando genérico já mostrado anteriormente. O comando de parametrização para este processo é então o que se encontra na tabela 13.

Tabela 13 – Parametrização do processo reva_prom_d_proc na *framework*.

Comando de parametrização do processo revf_com_deal_dot_neg_prom_proc
/opt/IBM/InformationServer/Server/DSEngine/bin/ds*job* -run -wait -mode RESET *FRAMEWORK* js_post_*job*.reva_prom_d_proc; /opt/IBM/InformationServer/Server/DSEngine/bin/ds*job* -run -*job*status -warn 0 \ -param *FRAMEWORK*="PRD" \ -param param_process="reva_prom_d" \ -param param_stage="proc" \ -param param_domain="profit_loss" \ -param param_process_type="profit_loss" \ -param param_deadline="" \ -param param_sub_domain="" \ -param param_status="ACTIVE" \ -param sourceFilePath="/opt/IBM/Projects/ds_files/profit_loss/*framework*/" \ -param sourceFilenameSources="sources_*Hadoop*" \ -param sourceFilenameDependencies="dependencies_reva_prom_d" \ *FRAMEWORK* js_post_*job*.reva_prom_d_proc

A tabela final que este processo alimenta foi designada por reva_prom_d e, à imagem da tabela do processo anterior, tem o intuito de ter já a estrutura mais semelhante possível à tabela final. Este é o processo mais complexo e consequentemente possui muitas tabelas intervenientes no processo. As tabelas que se deve realçar são:

- Revf_com_deal_dot_neg_prom;
- Slsa_reporterd_net_sale_d;
- Pnla_all_discounts;
- Pura_avg_cost_ratio;
- Dim_prom_location;
- Dim_prom_product.

Como todos os outros processos, este também recorre a outras dimensões e tabelas de parametrização. No anexo X pode-se ver a estrutura da tabela reva_prom_d bem como o *script* da sua criação. Após a realização da primeira e segunda etapas, segue-se o

desenvolvimento em *datastage*. Na figura 41 pode-se visualizar a estrutura de pastas definidas para este processo.

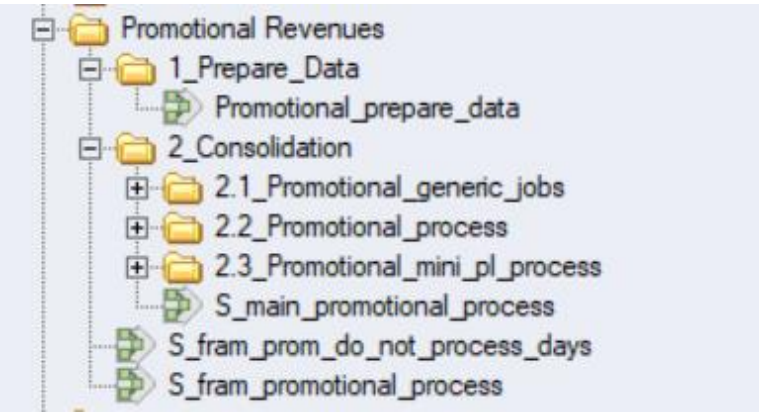


Figura 41 - Estrutura de pasta no *Datastage* do processo reva_prom_d.

A estrutura definida para este processo, devido à sua complexidade, como se pôde verificar na figura 41, é um pouco diferente. De forma a perceber esta estrutura, pode-se afirmar que o desenvolvimento em *datastage* deste processo pode ser divido em 5 grandes etapas. Sendo que estas etapas são as seguintes:

- Preparação das tabelas a utilizar para o *working schema*;
- Preparação das receitas – explosão das receitas que são cadastradas a um nível diferente do artigo, baseado nas tabelas de compras;
- Preparação da informação de venda/descontos dos últimos 30 dias tendo em conta apenas os artigos e localizações definidas para a promoção que está associada à receita;
- Distribuição das receitas;
- Inserção na tabela final reva_prom_d;

Seguindo a metodologia dos outros processos o primeiro *job* é o S_fram_promotional_process. Este *job* consiste num *job* quase genérico uma vez que com mais ou menos lógica irá fazer sempre as mesmas ações.

Este *job* irá verificar os dias que as dependências processaram invocar e injetar esses mesmos dias no *job* S_main_marginal_process e no final da execução deste último referido escrever na *framework* os dias que acabou de processar. No entanto este processo já no primeiro *job* é um pouco diferente.

Além de efetuar o que já foi referido, valida se existem receitas que tenham de ser distribuídas posteriormente. Isto deve-se ao facto de poder haver receitas com data de fim de promoção posterior à execução não tendo assim informação de vendas ou descontos. Para resolver isto, o seuqence *job* S_fram_promotional_process valida se existem receitas a ser distribuídas posteriormente e, caso haja esses dias, são injetados num *job* que unicamente abre a sua execução, escreve os dias em que as receitas que irão ser distribuídas posteriormente foram cadastradas, e termina a execução. Este *job* designa-se de S_fram_prom_do_not_process_days. Importante dizer que este último é uma dependência do processo de receitas de investimento promocional permitindo assim validar se essas receitas estão prontas a ser distribuídas nas execuções futuras.

O *sequence job* que se segue é o S_main_promotional_process. As imagens das figuras 42, 43 e 44 ilustram a estrutura do *job* onde cada componente, ou conjunto de componentes possui um número identificador. Para cada número elencou-se uma descrição sendo a mesma, a descrição da funcionalidade de cada componente ou conjunto de componentes.

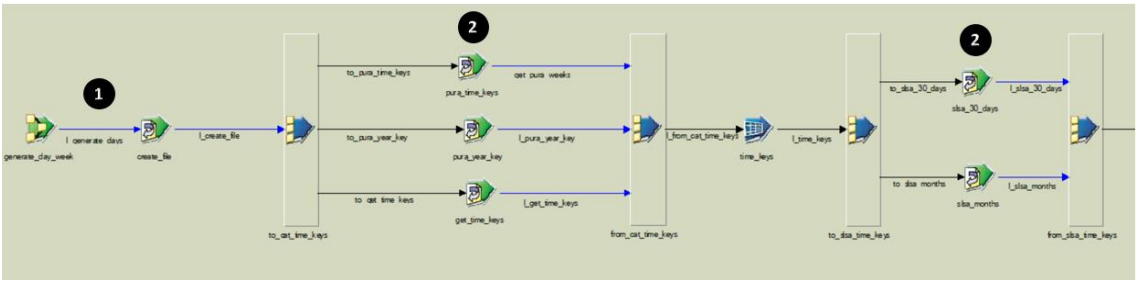


Figura 42 - *Sequence job* - S_main_promotional_process – parte 1/3.

1. Materializa num ficheiro a 2ª feira da semana respetiva ao *time_key* recebido;
2. Calcula os conjuntos de datas para processamento, com base nas *time_keys* recebidos para usar no processo;

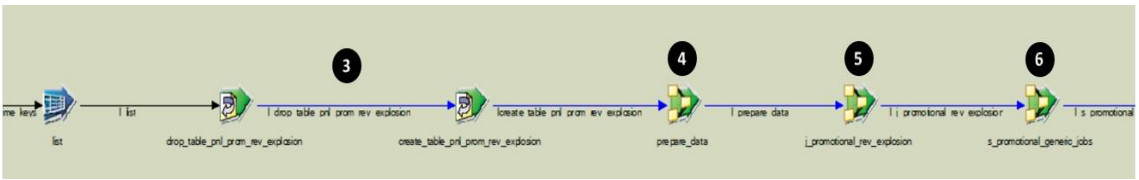


Figura 43 - *Sequence job* - S_main_promotional_process – parte 2/3.

3. Executa as *query*s responsáveis pelo *drop*/*create* das tabelas *working*

4. Executa o *sequence job* responsável pela preparação de dados para o processo de receitas promocionais;
5. [j_promotional_rev_explosion] Executa o *parallel job* responsável pela explosão das receitas marginais ao artigo;
6. [S_promotional_generic_*job*s] Executa o *sequence job* responsável pelo processamento dos *job*s genéricos para as receitas marginais;

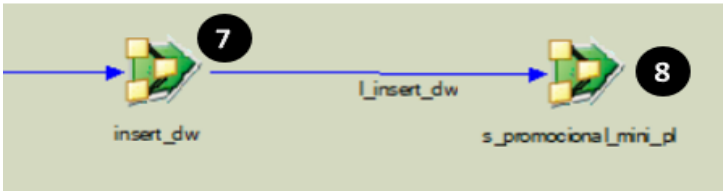


Figura 44 - *Sequence job* - S_main_promotional_process – parte 3/3.

7. [Execute_Beeline] Executa a *query* responsável pelo *merge* na tabela final, reva_prom_d;
8. [S_promocional_mini_pl] Invoca o processo que efetua a agregação e respetiva inserção na tabela pnla_analysis_d;

Como o processo anterior este processo também possui 5 grandes etapas de implementação. As etapas de preparação das tabelas a utilizar para o *working schema* e de desdobramento das receitas que são cadastradas a um nível diferente do artigo, baseado nas tabelas de compras, do processo de distribuição de receitas de investimento promocional é igual ao processo de distribuição de receitas marginais.

A etapa de preparação da informação de venda/descontos dos últimos 30 dias tendo em conta apenas os artigos e localizações definidas para a promoção que está associada à receita é a grande diferença. Nas figuras 45, 46, 47 e 48 pode-se visualizar essas mesmas diferenças.

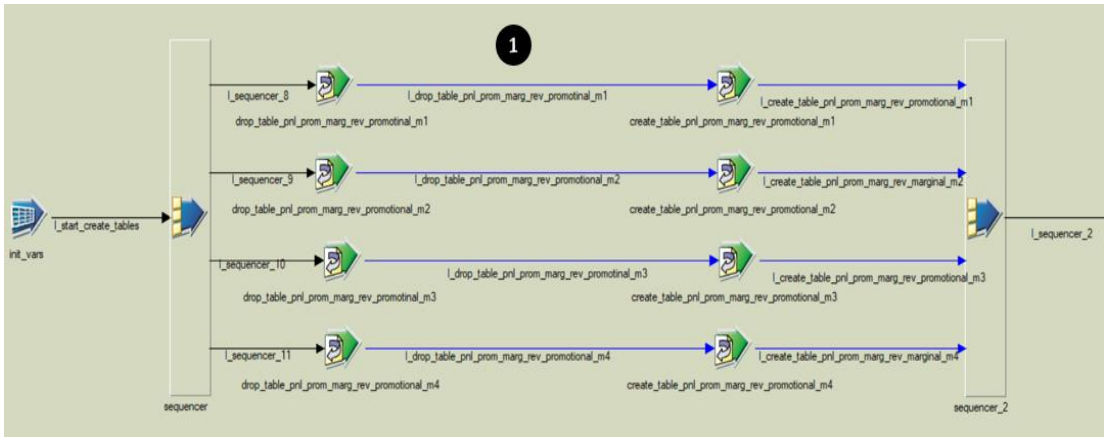


Figura 45 - *Sequence job* - S_promotional_process – parte 1/4.

1. Executa as *query*s responsáveis pelo drop/create das tabelas working;

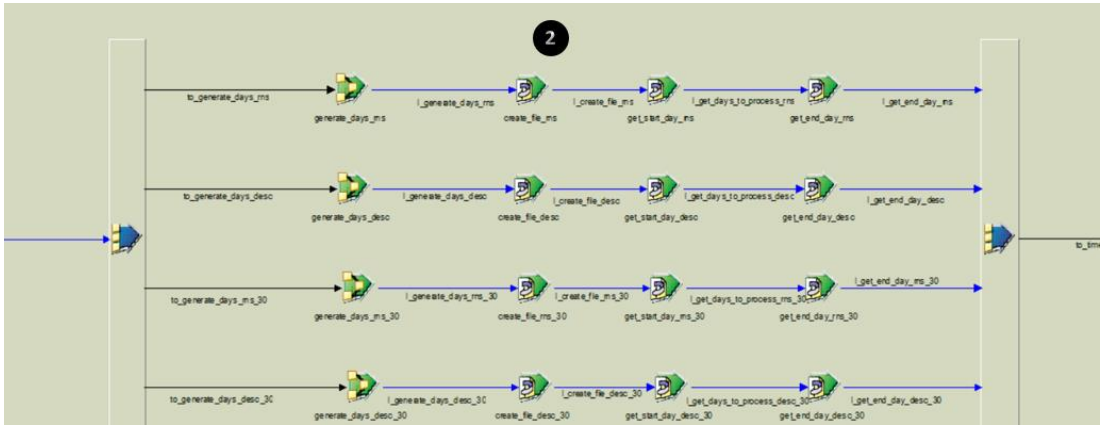


Figura 46 - *Sequence job* - S_promotional_process – parte 2/4.

2. Materializa num ficheiro as datas de início e fim de promoção;

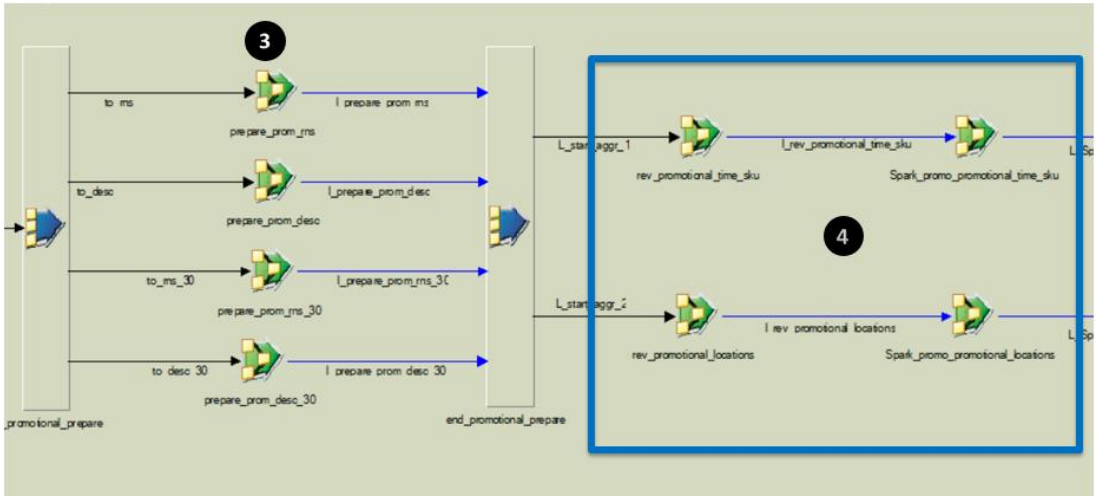


Figura 47 - *Sequence job* - S_promotional_process – parte 3/4.

Na figura 47, o número 3 materializa a tabela de vendas e descontos no intervalo temporal total. O número 4 representa então a grande diferença deste processo pois é nesta fase que se implementa a utilização de *Spark*. Os componentes mais à esquerda no número 4 criam tabelas no *working schema* vazias, enquanto os mais à direita invocam os *sequence job*s que despoletam a utilização de *Spark* no processo. Tal utilização deve-se ao facto de as tabelas dim_prom_product e dim_prom_location possuírem um enorme volume de dados.

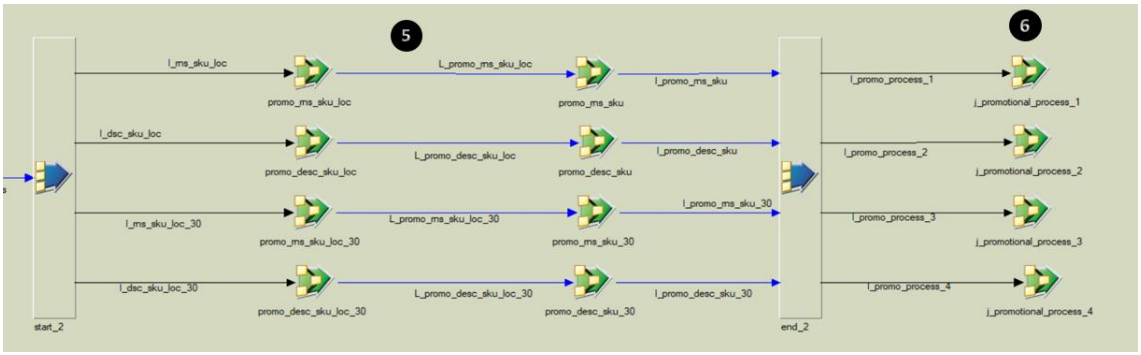


Figura 48 - *Sequence job* - S_promotional_process – parte 4/4.

5. Materializa as tabelas com os diferentes níveis de agregação de vendas e descontos por promoção. Os artigos e localizações são filtradas pela informação que resulta dos processos de *Spark*.

6. [J_promotional_process_1,2,3,4] Executa os *parallel job* responsável pelos dois diferentes processos de receitas promocionais;

Os processos de distribuição das receitas de investimento promocional são semelhantes aos processos de distribuição de receitas marginais. A única diferença é o intervalo temporal de rateio sendo que na distribuição de receitas de investimento promocional a receita é distribuída pelo período da campanha ou no dia de negociação em que a receita é cadastrada. De forma quase obrigatória uma vez que as estruturas das tabelas do processo de receitas marginais e do processo de receitas de investimento promocional serem iguais a inserção na tabela final também é igual mudando apenas o número de tabelas intervenientes nessa inserção.

Posto isto a diferença mais evidente deste processo para o processo para o de receitas marginais é a utilização de *Spark*. Foi necessário recorrer as duas vezes a esta tecnologia. Na figura 49, pode visualizar-se um dos *job*s que invocam a utilização do *Spark*, pois são os dois

iguais exceto o ficheiro que invocam. Esta tecnologia recorre a ficheiros com extensão *scala*. No anexo XI pode-se visualizar o conteúdo dos ficheiros utilizados.

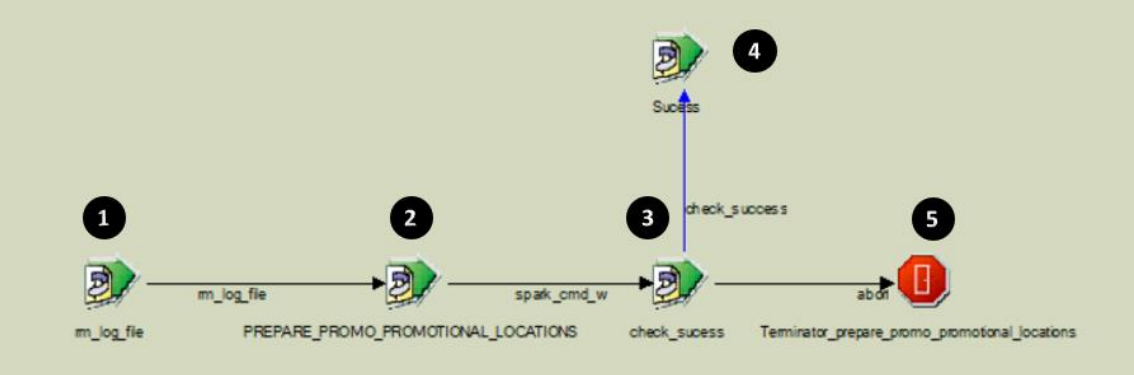


Figura 49 - *Sequence job* - S_prepare_promo_promotional_locations.

Como se pode visualizar este *sequence job* não é nada complexo a nível de componentes e lógica pois a complexidade reside na invocação da tecnologia e respetivo ficheiro. De modo muito simples, o componente identificado com o número 1 apaga o conteúdo do ficheiro que guarda os *logs* de cada execução, o número 3 efetua a validação da execução, o número 4 caso emite a mensagem de sucesso caso tudo corra em conformidade. Caso contrário, o componente número 5 é chamado a intervir abortando o processo todo.

No componente número 2 reside toda a complexidade, pois é onde se define o paralelismo a utilizar, a memória a alocar, o ficheiro a utilizar, entres outros parâmetros relevantes. Na tabela 14 encontra-se o comando utilizado.

Tabela 14 - Comando de invocação de *Spark*.

Comando de invocação de Spark
cp #PROJECT_FOLDERS.AUX_FILES#spark/spark_promo_promotional_locations.scala #PROJECT_FOLDERS.AUX_FILES#/spark/tmp/spark_promo_promotional_locations && sed -i -e 's/\[SCHEMA_CORE\]/#*SCHEMAS*.CORE#/g' #PROJECT_FOLDERS.AUX_FILES#spark/tmp/spark_promo_promotional_locations && sed -i -e 's/\[SCHEMA_WORKING\]/#*SCHEMAS*.WORKING#/g' #PROJECT_FOLDERS.AUX_FILES#spark/tmp/spark_promo_promotional_locations && spark2-shell --name "PREPARE PROMO PROMOTIONAL LOCATIONS" --queue=#POOL.POOL_NAME# --conf spark.sql.hive.caseSensitiveInferenceMode=NEVER_INFER --conf spark.sql.shuffle.partitions=100 --conf spark.dynamicAllocation.enabled=false --executor-memory 10g --num-executors 19 spark.yarn.executor.memoryOverhead=1.0g -i #PROJECT_FOLDERS.AUX_FILES#spark/tmp/spark_promo_promotional_locations > #PROJECT_FOLDERS.AUX_FILES#spark/spark_promo_promotional_locations_log

5.4. PROCESSO DE AGREGAÇÃO E INSERÇÃO NA P&L COMERCIAL

Este é o último processo sendo que é comum quer ao processo de receitas marginais quer ao de receitas de investimento promocional. Este processo não possui *framework* uma vez que se encontra inserido nos processos anteriores. Na figura 50, pode-se verificar a sua localização na estrutura de pastas dos dois processos.

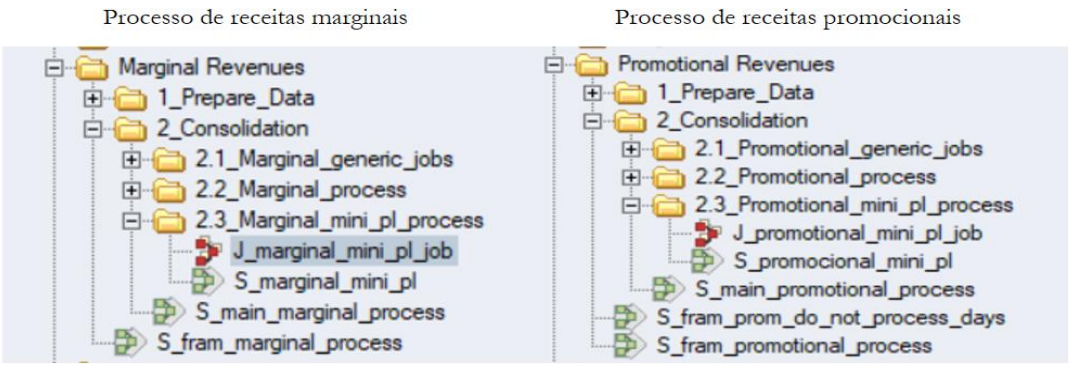


Figura 50 - Localização na estrutura de pastas nos dois processos do processo de inserção na P&L Comercial.

Este processo de inserção na tabela pnla_analysis_d é idêntico nos dois processos de distribuição. Posto isto, nas próximas figuras tem-se a explicação da implementação de um deles. Importante dizer que este processo possui um *sequence job* e um *parallel job*. Na figura 51, tem-se um dos *sequence jobs* implementados.

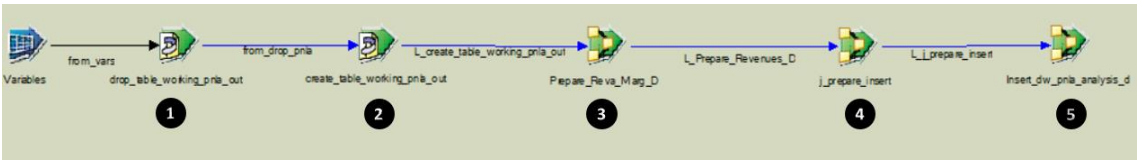


Figura 51 - *Sequence job* - *Sequence* do processo de inserção na tabela de pnla_analysis_d [Processo de receitas marginais].

1. Efetua drop à tabela no schema working [pnl_marg_rev_pnl_analysis_d];
2. Executa o comando de *create table* no *schema working* [pnl_marg_rev_pnl_analysis_d];
3. Executa o *prepare data* da tabela dw.reva_marg_d para o *working schema*;
4. Invocação do *job* paralelo que efetua a agregação para inserção da informação na tabela pnla_analyis_d;
5. Executa o comando de *insert* na tabela do *schema DW* [Pnla_analysis_d];

Após o *sequence* efetuar o *drop* e *create* da tabela de *working* inicia-se então o *parallel job*. Nas figuras 52 e 53 pode-se visualizar a implementação da lógica de agregação.

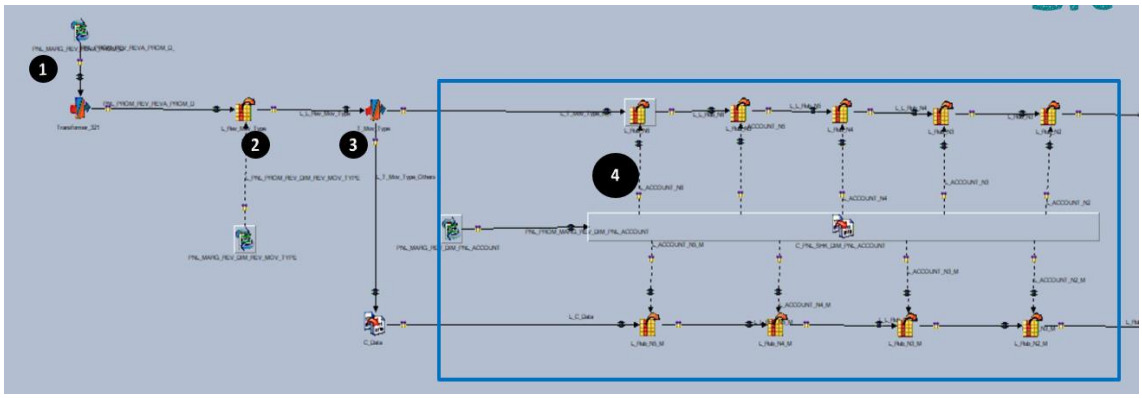


Figura 52 - *Parallel job* - *Parallel* do processo de inserção na tabela de pnla_analysis_d [Processo de receitas marginais] – parte 1/2.

1. Carrega os dados das tabelas necessárias ao processo;
2. Efetua *join* com a dim_mov_type para saber o mov_type_cd
3. Divide a informação em receitas marginais e em receitas promocionais com tratamento igual a receitas de investimento marginal.
4. Processo dinâmico de busca da *account_key* (desde de nível 6 até nível 2);

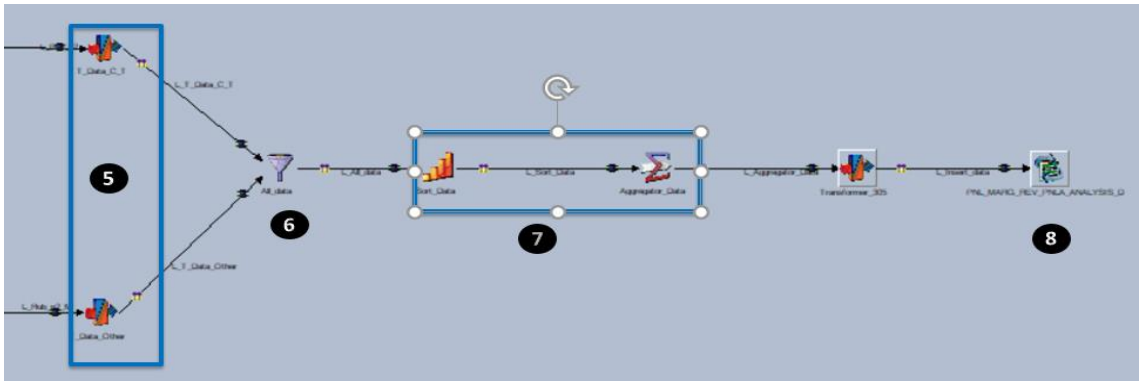


Figura 53 - *Parallel job* - *Parallel* do processo de inserção na tabela de pnla_analysis_d [Processo de receitas marginais] – parte 2/2.

5. Transforma os dados para a estrutura final;
6. Une a informação dos dois ramos;
7. Ordenação e agregação dos dados por dia, localização, produto, *account_key*, *account_domain_key* e *currency*;
8. Inserção na tabela final no *working schema* [pnl_marg_rev_pnl_analysis_d];

Como ação final deste processo é realizada a inserção na tabela pnla_analysis_d. Na tabela 15, tem-se o *script* de inserção na tabela final, chegando assim o final e atingido o pressuposto inicial.

Tabela 15 - Insert final na tabela Pnla_Analysis_D.

Insert final na tabela Pnla_Analysis_D
SET hive.exec.parallel=TRUE; SET hive.exec.dynamic.partition=TRUE; SET hive.exec.dynamic.partition.mode=nonstrict; INSERT OVERWRITE TABLE ":*SCHEMAS*.CORE:".PNLA_ANALYSIS_D PARTITION (day_key, account_domain_key) SELECT a.time_key, a.product_key, a.sku, a.sku_aggr_key, a.location_key, a.location_cd, a.loc_aggr_key, a.account_key, a.currency_key, a.system_key, a.amount, a.amount_eur, a.create_date, a.last_updt_date, a.time_key AS day_key, a.account_domain_key FROM ":*SCHEMAS*.WORKING:".PNL_PROM_REV_PNLA_ANALYSIS_D a WHERE a.time_key IN (":list.DAYS_PROCESSED:") AND a.account_domain_key = (":PROMOTIONAL_REV_PARAMS.ACC_DOMAIN_KEY:") UNION ALL SELECT b.time_key, b.product_key, b.sku, b.sku_aggr_key, b.location_key, b.location_cd, b.loc_aggr_key, b.account_key, b.currency_key, b.system_key, b.amount, b.amount_eur, b.create_date, b.last_updt_date, b.day_key, b.account_domain_key FROM ":*SCHEMAS*.CORE:".PNLA_ANALYSIS_D b WHERE b.day_key IN (":list.DAYS_PROCESSED:") AND b.account_domain_key = (":PROMOTIONAL_REV_PARAMS.ACC_DOMAIN_KEY:") AND b.account_key NOT IN (":PROMOTIONAL_REV_PARAMS.ACC_KEY_LIST:");

5.5. VALIDAÇÃO DOS RESULTADOS OBTIDOS

A validação da implementação e demonstração dos resultados são claramente o que dá segurança a quem irá utilizar esta informação. Pretende-se que tudo o que foi desenvolvido responda às necessidades existentes.

A primeira validação a efetuar do processo que passava por juntar a informação de receitas com a informação relativa a promoções. A melhor forma de demonstrar isso é validar para dois determinados registos um que deve possuir informação sobre promoções e outro não e verificar se isso acontece sem criar mais linhas uma vez que se se possui uma única linha essa não deve ser duplicada. Nas figuras 54, 55 e 56 pode ver-se exatamente essa validação.

```
select *
from dw.revf_com_deal_dot_neg
WHERE month_key = 201807
AND time_key = 20180725
and location_key = 10010002
and supplier_key =  43129010001
and rev_type_key = 'CPROM010001'
and prod_hierarchy_key = 121301010001
and contract_no =   12233350
and sequence_no = 850739454;
```

Query History | Consultas guardadas | Resultados (1)

	revf_com_deal_dot_neg.time_key	revf_com_deal_dot_neg.system_key	revf_com_deal_dot_neg.location_key	revf_com_deal_dot_neg.supplier_key	revf_com_deal_dot_neg.rev_t
1	20180725	9	10010002	43129010001	CPROM010001

Row details

revf_com_deal_dot_neg.time_key	20180725
revf_com_deal_dot_neg.system_key	9
revf_com_deal_dot_neg.location_key	10010002
revf_com_deal_dot_neg.supplier_key	43129010001
revf_com_deal_dot_neg.rev_type_key	CPROM010001
revf_com_deal_dot_neg.offer_key	3163657010001
revf_com_deal_dot_neg.prod_hierarchy_key	121301010001
revf_com_deal_dot_neg.calc_type	C
revf_com_deal_dot_neg.head_seq_no	850738733
revf_com_deal_dot_neg.action_no	1
revf_com_deal_dot_neg.contract_no	12233350

Figura 54 – Tabela origem do processo de criação do novo modelo de receita.

```
select *
from dw.revf_com_deal_dot_neg_prom
WHERE time_key = 20180725
and location_key = 10010002
and supplier_key =  43129010001
and rev_type_key = 'CPROM010001'
and prod_hierarchy_key = 121301010001
and contract_no =   12233350
AND month_key = 201807
and sequence_no = 850739454;
```

Query History | Consultas guardadas | Resultados (1)

	revf_com_deal_dot_neg_prom.time_key	revf_com_deal_dot_neg_prom.location_key	revf_com_deal_dot_neg_prom.supplier_key	revf_com_deal_dot_neg_prom.rev_type_key	revf_com_deal_dot_neg
1	20180725	10010002	43129010001	CPROM010001	3163657

Figura 55 Tabela final do processo de criação do novo modelo de receita.

Row details

revf_com_deal_dot_neg_prom.time_key	20180725
revf_com_deal_dot_neg_prom.location_key	10010002
revf_com_deal_dot_neg_prom.supplier_key	43129010001
revf_com_deal_dot_neg_prom.rev_type_key	CPROM010001
revf_com_deal_dot_neg_prom.promotion_cd	3163657
revf_com_deal_dot_neg_prom.prod_hierarchy_key	121301010001
revf_com_deal_dot_neg_prom.contract_no	12233350
revf_com_deal_dot_neg_prom.sequence_no	850739454
revf_com_deal_dot_neg_prom.negotiation_date	2018-07-25 00:00:00.0
revf_com_deal_dot_neg_prom.cat_cd	1301
revf_com_deal_dot_neg_prom.subcat_cd	NULL
revf_com_deal_dot_neg_prom.brand_cd	NULL
revf_com_deal_dot_neg_prom.sku	NULL
revf_com_deal_dot_neg_prom.prom_start_date	2018-07-17 00:00:00.0
revf_com_deal_dot_neg_prom.prom_end_date	2018-07-23 00:00:00.0
revf_com_deal_dot_neg_prom.dif_neg_prom	2
revf_com_deal_dot_neg_prom.calc_type	P_NH_1
revf_com_deal_dot_neg_prom.prom_valid	Y

Figura 56 - Informação da promocional na tabela final do processo de criação do novo modelo de receita.

O exemplo anterior trata-se de uma receita que possui informação promocional. A diferença para uma que não possua este tipo de informação é que os atributos com finalidade de apresentar esses valores se encontram a *null*.

De forma, a demonstrar os resultados dos processos de distribuição de receitas quer marginais quer de investimento promocional, bem como os de inserção na tabela final, pnla_analysis_d, foi selecionado um dia e uma categoria de produtos conseguindo com isto tornar os resultados mais simples de compreensão.

Na figura 57 pode-se visualizar então a informação do valor dessa categoria de produtos que deve ser rateada.

```
1 SELECT sum(int_dot_val)
2 FROM dw.revf_com_deal_dot_neg_prom np
3  INNER JOIN dw.dim_location dl ON dl.location_key = np.location_key
4  INNER JOIN dw.dim_loc_hierarchy dlh ON dlh.loc_hierarchy_key = dl.loc_hierarchy_key
5                                      AND dlh.loc_brand_cd IN (302, 303, 143,403, 306, 305, 888, 400)
6 WHERE month_key = 201807
7   AND time_key = 20180725
8   AND cat_cd = 303
9   AND rev_type_key NOT IN ('CPROM010001','TACLI010001');
10
```

Query History | Consultas guardadas | Resultados (1)

	_c0
1	41000

Figura 57 - Valor de receita marginal de uma categoria anterior ao processo de rateio.

Após o método de rateio o valor terá se manter tendo apenas de ser distribuído ao nível do produto e localização. Na figura 58 tem-se essa validação na tabela reva_marg_d para o dia e categoria em questão, da mesma forma que na figura 59 se encontra a validação da tabela pnla_analysis_d mantendo também o valor original.

```
SELECT round(sum(int_dot_val),1)
FROM dw.reva_marg_d d
inner join dw.dim_product dp on dp.product_key  = d.product_key
INNER JOIN dw.dim_prod_hierarchy dph ON dph.prod_hierarchy_key = dp.prod_hierarchy_key
and dph.department_cd= 10 and dph.biz_unit_cd= 3 and dph.cat_cd= 3
WHERE d.month_key = 201807
  AND d.negotiation_time_key = 20180725
  AND d.rev_type_key NOT IN ('CPROM010001','TACLI010001');
```

Query History | Consultas guardadas | Resultados (1)

	_c0
1	41000

Figura 58 - Valor de receita marginal na tabela reva_marg_d depois do processo de rateio.

```
Select  round(sum(ad.amount),1)
  from dw.pnla_analysis_d ad
  inner join dw.dim_product dp on dp.product_key = ad.product_key
inner join dw.dim_prod_hierarchy dph on dp.prod_hierarchy_key = dph.prod_hierarchy_key
and dph.department_cd= 10 and dph.biz_unit_cd= 3 and dph.cat_cd= 3
INNER JOIN dw.dim_location dl ON dl.location_key = ad.location_key
INNER JOIN dw.dim_loc_hierarchy dlh ON dlh.loc_hierarchy_key = dl.loc_hierarchy_key
                                    AND dlh.loc_brand_cd IN (403,306, 305, 888, 400, 302, 303, 143)
INNER JOIN dw.dim_chain ch ON ch.chain_key = dl.chain_key AND ch.chain_cd = 'MCH'
  where ad.day_key =20180725
and ad.account_domain_key = '1102010001'
and ad.account_key in ( '1102020401010001',
 '1102020402010001',
 '1102020403010001',
 '1102020404010001');
```

Query History | Consultas guardadas | Resultados (1)

	_c0
1	41000

Figura 59 - Valor de receita marginal após o processo de inserção na tabela pnla_analysis_d.

Com isto pode validar-se que o valor não se perde apesar de ter sido rateado. Na figura 60 pode-se validar essa distribuição pois tem-se a perceção do número de registos iniciais, e que aumentará na tabela reva_marg_d. No entanto, na tabela pnl_analysis_d o número de registos deve manter-se.

```
SELECT all_tables.TABELA ,all_tables.NR_DE_REGISTOS
FROM (SELECT 'REVF_COM_DEAL_DOT_NEG_PROM' AS TABELA, count(1) as NR_DE_REGISTOS
FROM dw.revf_com_deal_dot_neg_prom np
 INNER JOIN dw.dim_location dl ON dl.location_key = np.location_key
 INNER JOIN dw.dim_loc_hierarchy dlh ON dlh.loc_hierarchy_key = dl.loc_hierarchy_key AND dlh.loc_brand_cd IN (302, 303, 143,403, 306, 305, 888, 400)
WHERE month_key = 201807 AND time_key = 20180725 AND cat_cd = 303 AND rev_type_key NOT IN ('CPROM010001','TACLI010001')
UNION ALL
Select 'REVA_MARG_D' AS TABELA, count(1) as NR_DE_REGISTOS
FROM dw.reva_marg_d d
INNER JOIN dw.dim_product dp on dp.product_key  = d.product_key
INNER JOIN dw.dim_prod_hierarchy dph ON dph.prod_hierarchy_key = dp.prod_hierarchy_key and dph.department_cd= 10 and dph.biz_unit_cd= 3 and dph.cat_cd= 3
WHERE d.month_key = 201807 AND d.negotiation_time_key = 20180725 AND d.rev_type_key NOT IN ('CPROM010001','TACLI010001')
UNION ALL
SELECT 'PNLA_ANALYSIS_D' AS TABELA, count(1) as NR_DE_REGISTOS
  FROM dw.pnla_analysis_d ad
  INNER JOIN dw.dim_product dp on dp.product_key = ad.product_key
INNER JOIN dw.dim_prod_hierarchy dph on dp.prod_hierarchy_key = dph.prod_hierarchy_key
and dph.department_cd= 10 and dph.biz_unit_cd= 3 and dph.cat_cd= 3
INNER JOIN dw.dim_location dl ON dl.location_key = ad.location_key
INNER JOIN dw.dim_loc_hierarchy dlh ON dlh.loc_hierarchy_key = dl.loc_hierarchy_key AND dlh.loc_brand_cd IN (403,306, 305, 888, 400, 302, 303, 143)
INNER JOIN dw.dim_chain ch ON ch.chain_key = dl.chain_key AND ch.chain_cd = 'MCH'
WHERE ad.day_key =20180725 and ad.account_domain_key = '1102010001'
and ad.account_key in ( '1102020401010001', '1102020402010001', '1102020403010001', '1102020404010001')) all_tables;
```

Query History | Consultas guardadas | Resultados (9)

	all_tables.tabela	all_tables.nr_de_registos
1	REVF_COM_DEAL_DOT_NEG_PROM	32
2	PNLA_ANALYSIS_D	19598
3	REVA_MARG_D	19598

Figura 60 - Número de registos desde o início até ao final do processo de rateio e inserção.

O exemplo utilizado foi de receitas marginais, no entanto, transmite confiança para os processos de receitas promocionais uma vez que o processo de rateio é muito semelhante tendo apenas a restrições dos produtos e localizações.

Numa implementação deve ter-se também presente os tempos de execução face a um determinado número de registos.

Na tabela 16 encontra-se a informação referente ao desempenho dos processos, no que diz respeito à informação de uma determinada semana.

Tabela 16 - Desempenho global do processo.

Desempenho global do processo		
Processo	**Número de registos**	**Tempo de execução**
Processo de criação do novo modelo de receita	245.647	≈ 10 min.
Processo de distribuição de receitas marginais	818.390	≈ 45 min.
Processo de distribuição de receitas de investimento promocional	1.139.977	≈ 45 min.
Processo de inserção de receitas marginais	818.390	≈ 15min.
Processo de inserção de receitas de investimento promocional	1.139.977	≈ 15 min.

O processo já existente demorava cerca de 5 horas a processar a informação mensal. Partindo deste como termo de comparação, constata-se que a informação da nova estrutura permite possuir um volume muito superior e uma complexidade superior, conseguindo simultaneamente tempos de processamento inferiores.

Logicamente que, numa fase inicial, os tempos não eram os que foram apresentados. Os primeiros tempos obtidos em alguns processos eram superiores na ordem dos 65%. Os processos de distribuição de receitas de investimento promocional e de receitas marginais demoravam aproximadamente 1 hora e 30 minutos.

Através de otimizações dos *parallel jobs*, da alocação de mais memória e *containers* para execução dos processos, bem como a alteração de algumas tarefas para *Spark*, entre outras, conseguiu-se obter tempos de processamento que se julgam aceitáveis.

6. CONCLUSÕES

Esta dissertação destinou-se ao desenvolvimento de processos em ferramentas de *Big Data* para auxiliar a tomada de decisão. Para tal, esta implementação apoiou-se num conjunto de requisitos e especificações que levaram à definição dos processos a implementar.

Para elaborar uma solução coesa e eficaz é necessário ter-se sempre presentes os conceitos de *data warehousing*, modelação multidimensional e acima de tudo processos ETL, conceitos relacionas com *Business Intelligence*, que se encontram no capítulo 2. Estes conceitos, tiveram um papel importante de sustentação teórico dos processos implementados e na sua idealização.

As ferramentas e a caracterização efetuada no capítulo 3, sobre *Big Data, Hadoop* e *Spark*, estes dois últimos, modelos de programação para processamento de grandes volumes de dados, utilizados no desenvolvimento dos processos desenvolvidos, também tiveram uma contribuição na perceção de qual o comportamento das aplicações e como de deve trabalhar grandes volumes de dados. Toda a sustentação teórica ajudou a uma boa implementação.

Pessoalmente considerei este projeto de dissertação bastante desafiador e interessante, pois confere uma visão das necessidades atuais das organizações e do uso de tecnologias recentes e com grande capacidade de processamento e auxílio da tomada de decisão.

Sendo desafiador, foi ao mesmo tempo difícil, encontrando como maior dificuldade fazer a ponte entre a linguagem que as direções comerciais utilizam e a linguagem que a uma área técnica utiliza de forma a alinhar os requisitos de forma a desenvolver uma solução

robusta, eficaz e escalável. A utilização de ferramentas atuais também foi bastante enriquecedor e desafiador.

A dificuldade imposta nos algoritmos de distribuição de receitas marginais e receitas de investimento promocional serviram para ter uma perceção de como agir face a um problema gigante partindo-o em problemas mais pequenos e de mais fácil resolução.

Como é obvio, existe dois aspetos a melhor a performance dos processos e a distribuição equitativa realizada nos processos de distribuição.

Em todas as soluções se deve perceber e procurar onde poderão estar os pontos de falha e consequentemente tentar mitiga-las. A melhoria continua deve então estar sempre presente no dia-a-dia de quem desenvolve e implementa uma solução qualquer que seja as necessidades a que responde.

Quanto à melhoria de distribuição equitativa dos processos de distribuição de receitas, tem que ver com o fato de neste momento estar a ser feito uma distribuição cega sem ter noção dos tipos de produtos, ou seja, podermos estar a atribuir valor de receita de um determinado tipo de produto a uma localização que não o vende.

Em suma, este estudo contribui para um enriquecimento pessoal na área de *Business Intelligence* e *Big Data* quer a nível de conceitos quer a nível de ferramentas. Contudo, este estudo salienta que, apesar de muitas vezes as áreas de sistemas e de tecnologias de informação parecerem distantes da vida das organizações, proporcionam através dos processos que desenvolvem, uma base de sustentação á tomada de decisão. Com isto e é cumprido assim o intuito do *Business Intelligence* e do *Big Data*, acrescentar valor ás organizações.

7. REFERÊNCIAS DOCUMENTAIS

Akerkar, R. (2014). Big Data Computing, CRC Press.

Alter, S. (2002). A Work System View of DSS in its Fourth Decades. Americas Conference on Information Systems (AMCIS), Dallas.

Arias, J., J. A. Gamez and J. M. Puerta (2016). "Learning distributed discrete Bayesian Network Classifiers under MapReduce with Apache SparkFrom Databases to Big Data." Knowledge-Based Systems **117**: 16-26.

Chen, G., S. Wu and Y. Wang (2015). "The evolvement of big data systems: from the perspective of an information security application." Big Data Research **2**(2): 65-73.

Cody, W. F., J. T. K. Kreulen, V. Krishna and W. S. Spangler (2002). "The integration of business intelligence and knowledge management." IBM Systems Journal **41**(4): 697-713.

Côrte-Real, N., T. Oliveira and P. Ruivo (2017). "Assessing business value of Big Data Analytics in European firms." Journal of Business Research **70**: 379-390.

Davenport, T. H. (2006). Competing on Analytics. Harvard Business Review. Massachusetts, Harvard Business Publishing. **84:** 98-107.

Dumbill, E. (2012). "What is big data? An introduction to the big data landscape." O'reilly Media.

Elisabeth, I.-Z., E. Anikó, K. Zsolt, B. Christopher, W. Philip and M. Laszlo (2015). "Advanced predictive-analysis based decision support for collaborative logistics networks." Supply Chain Management: An International Journal **20**: 369-388.

Gaboleiro, A. (2016). "BIT, uma marca que quer criar experiências no retalho." Retrieved October 17, 2018, from http://imagensdemarca.sapo.pt/atualidade/bit-uma-marca-que-quer-criar-experiencias-no-retalho/.

Gardner, S. R. (1998). Building the Data Warehouse. Communications of the ACM. New York, USA. **41:** 52-60.

Han, J. and M. Kamber (2001). Data Mining: Concepts and Techniques. San Francisco, CA, USA, Morgan Kaufmann.

Inmon, W. H. (1996). The Data Warehouse and Data Mining. Communications of the ACM New York. **39:** 49-50.

Inmon, W. H. (2002). Building the Data Warehouse. USA, John Wiley & Sons, Inc.

Jain, V. (2017). "Bridging two worlds : Integration of SAP and Hadoop Ecosystems." Retrieved 18 Oct, 2018, from https://blogs.sap.com/2017/07/19/bridging-two-worlds-integration-of-sap-and-hadoop-ecosystems/.

Kimball, R. (1996). The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses, John Wiey & Sons, Inc.

Kimball, R. and J. Caserta (2004). The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. Indianapolis, John Wiley & Sons, Inc.

Kimball, R., L. Reeves, M. Ross and W. Thornthwaite (1998). The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses, John Wiley & Sons, Inc.

Kimball, R. and M. Ross (2013). The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling (Third edition). Indianopolis, John Wiley & Sons, Inc.

Maitrey, S. and C. K. Jha (2015). "MapReduce: Simplified Data Analysis of Big Data " Procedia Computer Science **57**: 563-571.

Manyika, J. (2011). Big data: the next frontier for innovation, competition, and productivity, McKinsey Global Institute.

Moody, D. L. and M. A. R. Kortink (2003). "From ER Models to Dimensional Models: Bridging the Gap between OLTP and OLAP Design, Part I." Business Intelligence Journal **8**(3): 7-24.

Pires da Silva, E. P. d. S., Ana Cristina (2010). SNC - Manual de Contabilidade, Rei dos Livros.

Ramachandramurthy, S., S. Subramaniam and C. Ramasamy (2015). "Distilling big data: refining quality information in the era of Yottabytes." The Scientific World Journal: 1-9.

Reynolds, H. (2016). "Cognitive computing Big data and cognitive computing–Part 2." KmWorld **25**(2): 2.

Sonae. (2018). Retrieved October 10, 2018, from https://www.sonae.pt/pt/investidores/portefolio/.

Vassiliadis, P., A. Simitsis and S. Skiadopoulos (2002). Conceptual Modeling for ETL Processes. DOLAP 2002 - 5th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, Vrigina, USA.

Yu, N. (2015). Big data analytics in power distribution systems. Power and energy society innovative smart grid technologies conference. Washington**:** 1-5.

8. ANEXOS

ANEXO I – Resumo das estratégias propostas por Kimball e Ross (2013) para alterações de SCD

Tipo SCD	Ação nos atributos	Impacto na análise
Tipo 0	Nenhuma alteração é aplicada.	Factos estão sempre associados a valores originais.
Tipo 1	Reescrever o valor dos atributos.	Factos associados aos valores atuais dos atributos.
Tipo 2	Inserção de novo registo.	Factos associados aos valores dos atributos a quando da sua ocorrência.
Tipo 3	Atributos adicionais na dimensão.	Factos associados aos valores atuais e passados.
Tipo 4	Nova tabela de dimensão, mais pequena, com os atributos que podem estar em constante mutação.	Factos associados aos valores dos atributos de alteração rápida a quando da sua ocorrência.
Tipo 5	Nova tabela de dimensão, mais pequena, com os atributos que podem estar em constante mutação e do Tipo 1.	Factos associados aos valores dos atributos de alteração rápida a quando da sua ocorrência e dos valores atuais.
Tipo 6	Adicionar atributos do Tipo 1 a dimensões do Tipo 2.	Factos associados aos valores dos atributos a quando da sua ocorrência e dos valores atuais.
Tipo 7	Dimensões do Tipo 1 e Tipo 2.	Factos associados aos valores dos atributos a quando da sua ocorrência e dos valores atuais.

Tabela 17 - Tabela resumo das propostas de alteração de SCD. Elaboração própria, baseado em Kimball e Ross (2013)

ANEXO II – Explicação dos pontos de decisão e das ações dos dois algoritmos

Tabela 18 - Tabela descritiva dos pontos de decisão dos algoritmos.

Pontos de Decisão		
Decisão Fluxograma	**Designação**	**Descrição**
SKU?	Cadastro de Receita ao SKU	Visa identificar a forma mercadológica como foi cadastrada a receita.
VND?	Vendas	Para uma receita cadastrada ao (Fornecedor * Sku * Brand), valida para o conjunto de artigos resultante dos processos de identificação de artigos elegíveis, se existem vendas nos últimos 30 dias, em que a agregação ao artigo seja positiva com base no valor de venda liquida. Quando promocional o período em análise corresponde a duração da promoção associada a receita.
DESC?	Descontos	Para uma receita cadastrada ao (Fornecedor * Sku * Brand), valida para o conjunto de artigos resultante dos processos de identificação de artigos elegíveis, se existem descontos nos últimos 30 dias, em que a agregação ao artigo seja positiva com base no valor de desconto. Quando promocional o período em análise corresponde a duração da promoção associada a receita.
SKUS na Lista?	Artigos elegíeis	Para uma determinada receita não cadastrada ao Sku, avalia se existem artigos elegíveis para o conjunto (Fornecedor * Estrutura Mercadológica * Brand). Um artigo é considerado elegível após passar por todos os filtros a ele impostos pelo seu tipo de enquadramento de cadastro. Estes filtros são (Compras efetuadas nos últimos 15 ou 90 dias e/ ou associação a Promoção no caso de receitas promocionais).
VND na Lista?	Validação de Vendas para artigos associados por Compra\Promoção	Para uma receita cadastrada ao (Fornecedor * Estrutura Mercadológica * Brand), valida para o conjunto de artigos resultante dos processos de identificação de artigos elegíveis, se existem Vendas nos últimos 30

		dias, em que a agregação ao artigo seja positiva com base no valor de venda liquida. Quando promocional o período em análise corresponde a duração da promoção associada a receita.
DESC na Lista?	Validação de Descontos para artigos associados por Compra\Promoção	Para uma receita cadastrada ao (Fornecedor * Estrutura Mercadológica * Brand), valida para o conjunto de artigos resultante dos processos de identificação de artigos elegíveis, se existem Descontos nos últimos 30 dias, em que a agregação ao artigo seja positiva com base no valor de desconto. Quando promocional o período em análise corresponde a duração da promoção associada a receita.
VND Forn. Prior.?	Validação de Vendas ao Fornecedor Prioritário	Para uma receita cadastrada ao (Fornecedor * Estrutura Mercadológica * Brand), valida se existem vendas nos últimos 30 dias, em que a agregação ao artigo seja positiva com base no valor de venda liquida, em que o fornecedor da receita seja o prioritário. Quando promocional o período em análise corresponde a duração da promoção associada a receita.
DESC Forn. Prior.?	Validação de Desconto ao Fornecedor Prioritário	Para uma receita cadastrada ao (Fornecedor * Estrutura Mercadológica * Brand), valida para o conjunto de artigos resultante dos processos de identificação de artigos elegíveis, se existem descontos nos últimos 30 dias, em que a agregação ao artigo seja positiva com base no valor de desconto, para os descontos em que o fornecedor da receita seja o prioritário. Quando promocional o período em análise corresponde a duração da promoção associada a receita.

Tabela 19 - Tabela descritiva das ações dos algoritmos.

Ações	
Identificação da ação	**Descrição**
Calcular Pesos para 15 e 90 dias, se não houver taxa de 100% ao Artigo	Percentual de distribuição calculado com base em compras de 15 ou 90 dias para receitas cadastradas ao Sku. Quando relação entre artigo cadastrado e fornecedor não esteja presente na tabela de acordo com o seu cadastro a subcategoria, o valor aplicado é 100% de forma a não ser perdido o artigo associado a receita.
Compras ao Fornecedor nos últimos 15 e 90 dias.	Processo de desdobramento ao artigo para receitas cadastradas à estrutura hierárquica. Para uma receita cadastrada ao (Fornecedor * Estrutura Mercadológica * Brand) são identificados os artigos para os quais foram efetuadas compras nos últimos 15 e 90 dias, assim como o seu rácio de participação no artigo em questão.
Pesos para 15 e 90 dias	De acordo com o cadastro efetuado a subcategoria, são identificados os rácios a aplicar para cada um dos artigos resultantes do passo anterior.
Filtrar para artigos associados a Promoção	Filtro adicional para receitas do tipo promocional, onde serão excluídos os artigos não presentes na promoção associada a receita. O universo de lojas pela qual será efetuado o rateio, é também limitado as existentes na promoção.

ANEXO III – Fluxograma do algoritmo de receitas marginais

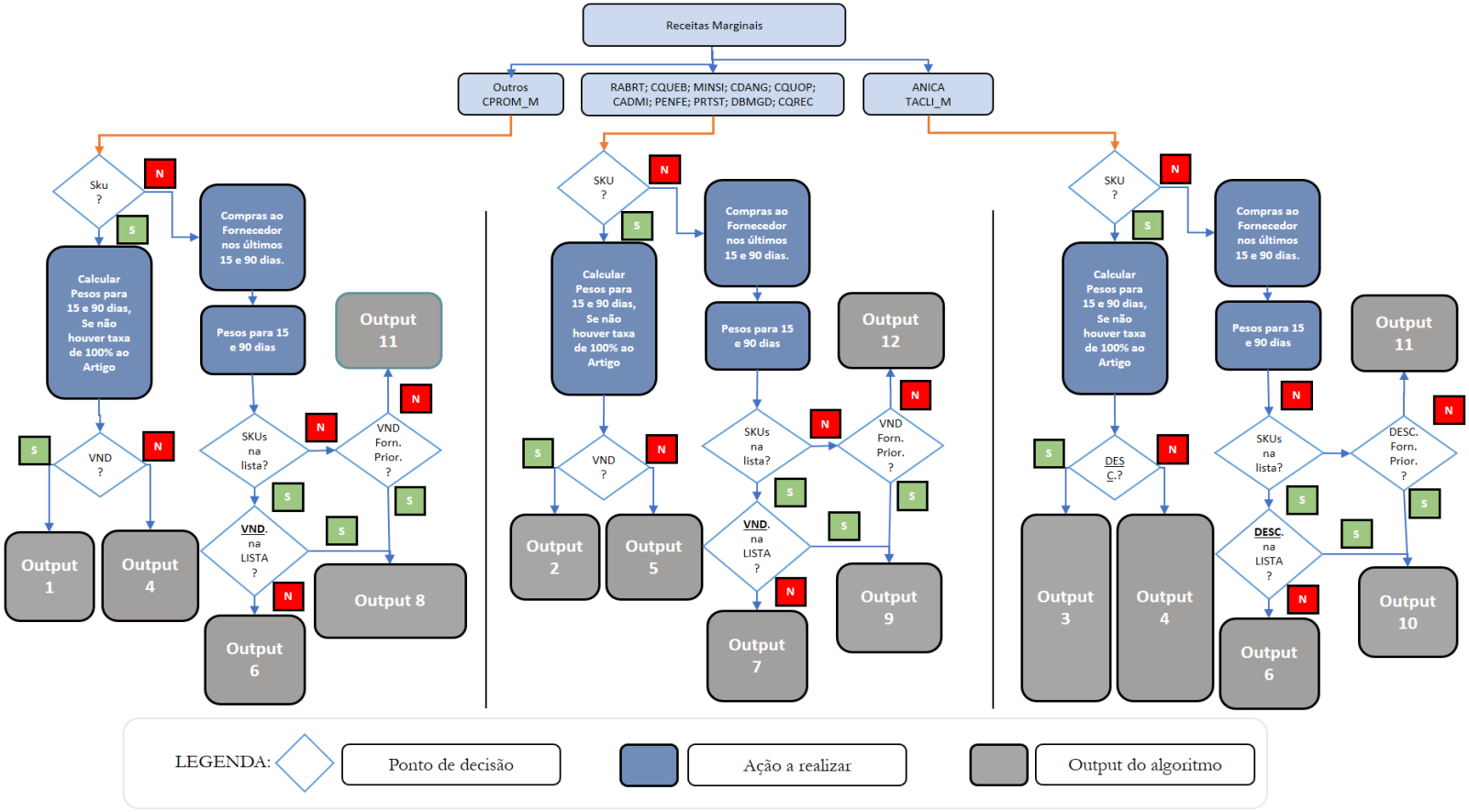


Figura 61 - Fluxograma do algoritmo de receitas marginais.

ANEXO IV – Output do algoritmo de receitas marginais

Tabela 20 - Outputs do algoritmo de receitas marginais.

Outputs Algoritmos de Receitas		
Identificação	**Designação**	**Descrição**
Output 1	Sku -Venda – Insígnias do Universo MCH	Rateio com base em venda, para todos os sku´s presentes na receita para as localizações pertencentes às seguintes Insígnias (I143, I302, I303, I305, I306, I400, I403, I888)
Output 2	Sku – Vendas – Lojas Receita	Rateio com base em venda por localização de Receita, para todos os sku´s presentes na receita para as localizações existentes na Receita
Output 3	Sku – Descontos - Insígnias do Universo MCH	Rateio com base em descontos, para todos os sku´s presentes na receita para as localizações pertencentes às seguintes Insígnias (I143, I302, I303, I305, I306, I400, I403, I888)
Output 4	Sku -Equitativa Skus da Receita – Insígnias do Universo MCH	Distribuir valor da receita de forma equitativa para todos os artigos cadastrado na receita para as lojas das seguintes insígnias (I143, I302, I303, I305, I306, I400, I403, I888)
Output 5	Sku – Equitativa Skus da Receita – Lojas Receita	Coloca valor da receita de forma por localização de Receita para todos os artigos cadastrado na receita. Neste caso não há rateio a loja é a da receita e os artigos também.
Output 6	Hierarquia Produto – Equitativa Lista de Artigos Compra - Insígnias do Universo MCH	Distribuir valor da receita de forma equitativa para todos os artigos presentes na lista de artigos comprados pelo fornecedor para as lojas das seguintes insígnias (I143, I302, I303, I305, I306, I400, I403, I888)
Output 7	Hierarquia Produto – Equitativa Lista de Artigos Compra – Loja Receita	Distribuir valor da receita de forma equitativa para todos os artigos presentes na lista de artigos comprados pelo fornecedor para as localizações existentes na receita.

Output 8	Hierarquia Produto – Vendas - Insígnias do Universo MCH	Ratear valor da receita com base em pesos de venda por todos os skus e lojas do universo MCH. Listagem de artigos a aplicar identificada com base em compras ou artigos em que o fornecedor seja o prioritário, para a (Estrutura Mercadológica * Brand) cadastrada na receita.
Output 9	Hierarquia Produto – Vendas – Loja Receita	Ratear valor da receita com base em pesos de venda por localização de receita por todos os skus. Listagem de artigos a aplicar identificada com base em compras ou artigos em que o fornecedor seja o prioritário, para a (Estrutura Mercadológica * Brand) cadastrada na receita.
Output 10	Hierarquia Produto – Descontos - Insígnias do Universo MCH	Ratear valor da receita com base em pesos de descontos por todos os skus e lojas do universo MCH. Listagem de artigos a aplicar identificada com base em compras ou artigos em que o fornecedor seja o prioritário, para a (Estrutura Mercadológica * Brand) cadastrada na receita.
Output 11	Hierarquia Produto – Equitativo Saco - Insígnias do Universo MCH	Distribuir valor da receita de forma equitativa para um artigo saco representante da hierarquia de cadastro para as lojas das seguintes insígnias (I143, I302, I303, I305, I306, I400, I403, I888)
Output 12	Hierarquia Produto – Equitativo Saco – Loja Receita	Coloca valor associado à localização da receita num artigo saco representante da hierarquia de cadastro.

ANEXO V – Fluxograma do algoritmo de receitas investimento promocional.

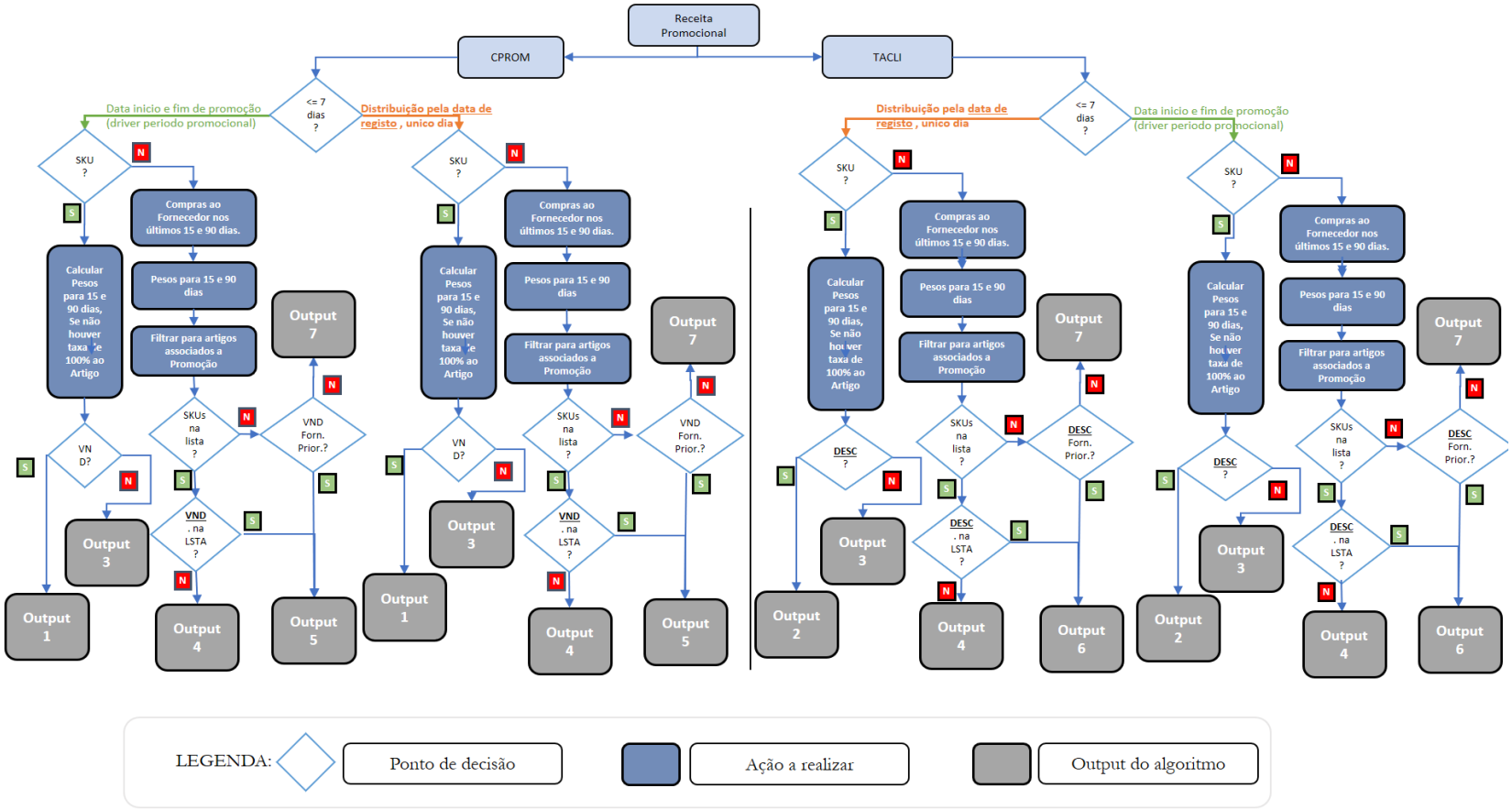


Figura 62 - Fluxograma do algoritmo de receitas promocionais.

ANEXO VI - Output do algoritmo de receitas investimento promocional

Tabela 21 - Outputs do algoritmo de receitas promocionais.

Outputs Algoritmos de Receitas		
Identificação	**Designação**	**Descrição**
Output 1	Sku -Venda – Insígnias do Universo MCH	Rateio com base em venda, para todos os sku´s presentes na receita para as localizações pertencentes às seguintes insígnias (I143, I302, I303, I305, I306, I400, I403, I888)
Output 2	Sku – Descontos - Insígnias do Universo MCH	Rateio com base em descontos, para todos os sku´s presentes na receita para as localizações pertencentes às seguintes insígnias (I143, I302, I303, I305, I306, I400, I403, I888)
Output 3	Sku -Equitativa Skus da Receita – Insígnias do Universo MCH	Distribuir valor da receita de forma equitativa para todos os artigos cadastrado na receita para as lojas das seguintes insígnias (I143, I302, I303, I305, I306, I400, I403, I888)
Output 4	Hierarquia Produto – Equitativa Lista de Artigos Compra - Insígnias do Universo MCH	Distribuir valor da receita de forma equitativa para todos os artigos presentes na lista de artigos comprados pelo fornecedor para as lojas das seguintes insígnias (I143, I302, I303, I305, I306, I400, I403, I888)
Output 5	Hierarquia Produto – Vendas - Insígnias do Universo MCH	Ratear valor da receita com base em pesos de venda por todos os skus e lojas do universo MCH. Listagem de artigos a aplicar identificada com base em compras ou artigos em que o fornecedor seja o prioritário, para a (Estrutura Mercadológica * Brand) cadastrada na receita.
Output 6	Hierarquia Produto – Descontos - Insígnias do Universo MCH	Ratear valor da receita com base em pesos de descontos por todos os skus e lojas do universo MCH. Listagem de artigos a aplicar identificada com base em compras ou artigos em que o fornecedor seja o prioritário, para a (Estrutura Mercadológica * Brand) cadastrada na receita.
Output 7	Hierarquia Produto – Equitativa Saco – Insígnias do Universo MCH	Distribuir valor da receita de forma equitativa para um artigo saco representante da hierarquia de cadastro para as lojas das seguintes Insígnias (I143, I302, I303, I305, I306, I400, I403, I888)

ANEXO VII – Estrutura da tabela do processo revf_com_deal_dot_neg_prom e *script* de criação

Tabela 22 - Estrutura da tabela revf_com_deal_dot_neg_prom.

Revf_com_deal_dot_neg_prom	
Atributo	**Tipo de Dados**
Time_key	Decimal (8,0)
Location_key	Decimal (38,0)
Supplier_key	Decimal (38,0)
Rev_type_key	Varchar (12)
Promotion_cd	Decimal (8,0)
Prod_hierarchy_key	Varchar (16)
Contract_no	Decimal (8,0)
Sequence_no	Decimal (10,0)
Negotiation_date	Timestamp
Int_dot_val	Decimal (20,4)
Int_curr_key	Varchar (9)
Sup_dot_val	Decimal (20,4)
Sup_curr_key	Varchar (9)
Biz_unit_cd	Decimal (4,0)
Cat_Cd	Decimal (4,0)
Subcat_cd	Decimal (4,0)
Brand_cd	Decimal (6,0)
Sku	Varchar (128)
Prom_start_date	Timestamp
Prom_end_date	Timestamp
Dif_neg_prom	Decimal (20,4)
Calc_type	Varchar (20)
Prom_valid	Varchar (1)
System_key	Decimal (38,0)
Create_date	Timestamp
Last_updt_date	Timestamp
Partição	
Month_key	Decimal (6,0)

Tabela 23 - *Script* de criação da tabela Revf_com_deal_dot_neg_prom.

***Script* de criação da tabela Revf_com_deal_dot_neg_prom**
DROP TABLE IF EXISTS DW.REVF_COM_DEAL_DOT_NEG_PROM;
CREATE EXTERNAL TABLE DW.REVF_COM_DEAL_DOT_NEG_PROM(
time_key decimal(8,0) COMMENT'Time Key of Negatiation',
location_key decimal(38,0) COMMENT'Location',
supplier_key decimal(38,0) COMMENT'Supplier',
rev_type_key varchar(12) COMMENT'Type of revenue',
promotion_cd decimal(8,0) COMMENT'Promotion',
prod_hierarchy_key varchar(16) COMMENT'Hierarchy Level',
contract_no decimal(8,0) COMMENT'Contract number',
negotiation_date timestamp COMMENT'Negotiation date',
int_dot_val decimal(20,4) COMMENT'Internal Value',
int_curr_key varchar(9) COMMENT'Internal currency',
sup_dot_val decimal(20,4) COMMENT'Supplier Value',
sup_curr_key varchar(9) COMMENT'Supplier Currency',
biz_unit_cd decimal(4,0) COMMENT'Biz Unit',
cat_cd decimal(4,0) COMMENT'Category',
subcat_cd decimal(4,0) COMMENT'Sub Category',
brand_cd decimal(6,0) COMMENT'Brand',
sku varchar(128) COMMENT'Product',
prom_start_date timestamp COMMENT'Start date of promotion',
prom_end_date timestamp COMMENT'End date of promotion',
dif_neg_prom decimal(20,4) COMMENT'Days between Negotiation date and End date of promotion',
calc_type varchar(20) COMMENT'Distribution type',
prom_valid varchar(1) COMMENT'Flag to valid promotion',
system_key decimal(38,0) COMMENT'System Key',
create_date timestamp COMMENT'Create date',
last_updt_date timestamp COMMENT'Last Update Date'
)
PARTITIONED BY (MONTH_KEY decimal(6,0) COMMENT'Key of partition - Month')
STORED AS PARQUET
LOCATION '/DATA/DW/CORE/REVF_COM_DEAL_DOT_NEG_PROM';

ANEXO VIII – Estrutura da tabela do processo de receitas marginais e *script* de criação

Tabela 24 - Estrutura da tabela reva_marg_d.

Reva_marg_d	
Atributo	**Tipo de Dados**
Endownment_time_key	Decimal (8,0)
Negotiation_time_key	Decimal (38,0)
Location_key	Varchar (128)
Location_cd	Varchar (128)
Loc_aggr_key	Decimal (38,0)
Product_key	Varchar (128)
Sku	Varchar (128)
Sku_aggr_key	Decimal (38,0)
Rev_type_key	Varchar (128)
Promotion_cd	Varchar (128)
Int_dot_val	Double
Int_curr_key	Varchar (9)
Sup_dot_val	Double
Sup_curr_key	Varchar (9)
System_key	Decimal (38,0)
Create_date	Timestamp
Last_updt_date	Timestamp
Partição	
Month_key	Decimal (6,0)

Tabela 25 - *Script* de criação da tabela reva_marg_d.

***Script* de criação da tabela reva_marg_d**

```
DROP TABLE IF EXISTS DW.REVA_MARG_D;
CREATE EXTERNAL TABLE DW.REVA_MARG_D
(
ENDOWNMENT_TIME_KEY  DECIMAL(8,0)     COMMENT 'DAY OF FACT',
NEGOTIATION_TIME_KEY DECIMAL(8,0)     COMMENT 'DAY OF FACT',
LOCATION_KEY         VARCHAR(128)     COMMENT 'LOCATION AGGREGATION KEY' ,
LOCATION_CD          VARCHAR(128)     COMMENT 'LCOATION CODE' ,
LOC_AGGR_KEY         DECIMAL(38,0)    COMMENT 'LOC AGGR KEY' ,
PRODUCT_KEY          VARCHAR(128)     COMMENT 'PRODUCT AGGREGATION KEY' ,
SKU                  VARCHAR(128)     COMMENT 'SKU' ,
SKU_AGGR_KEY         DECIMAL(38,0)    COMMENT 'SKU AGGR KEY' ,
REV_TYPE_KEY         VARCHAR(128)     COMMENT 'REV AGGREGATION KEY' ,
PROMOTION_CD         VARCHAR(128)     COMMENT 'PROMOTION AGGREGATION KEY' ,
INT_DOT_VAL          DOUBLE           COMMENT 'INTERNAL VALUE',
INT_CURRENCY_KEY     VARCHAR(9)       COMMENT 'INTERNAL CURRENCY',
SUP_DOT_VAL          DOUBLE           COMMENT 'SUPPLIER VALUE',
SUP_CURRENCY_KEY     VARCHAR(9)       COMMENT 'SUPPLIER CURRENCY',
SYSTEM_KEY           DECIMAL(38,0)    COMMENT 'SYSTEM KEY',
CREATE_DATE          TIMESTAMP        COMMENT 'CREATE DATE',
LAST_UPDT_DATE       TIMESTAMP        COMMENT 'LAST UPDATE DATE'
)
PARTITIONED BY (MONTH_KEY DECIMAL(6,0) COMMENT'MONTH OF FACT AND PARTITION')
STORED AS PARQUET LOCATION '/DATA/DW/CORE/REVA_MARG_D';
```

ANEXO IX – *Script* de inserção na tabela reva_marg_d

Insert final na tabela Pnla_Analysis_D

```
SET hive.exec.dynamic.partition=TRUE; SET hive.exec.dynamic.partition.mode=nonstrict; SET hive.exec.parallel = TRUE;
INSERT OVERWRITE TABLE ":SCHEMAS.CORE:".REVA_MARG_D partition (month_key)
SELECT m1.endownment_time_key, m1.time_key AS negotiation_time_key, dl.location_key,
m1.location_cd, m1.loc_aggr_key,dp.product_key, m1.sku,m1.sku_aggr_key,mt.mov_type_key AS rev_type_key,
m1.promotion_key AS promotion_cd,m1.int_dot_val,m1.int_curr_key AS int_currency_key,m1.sup_dot_val,m1.sup_curr_key AS sup_currency_key
,m1.system_key,m1.create_date,m1.last_updt_date, substr(m1.time_key,0,6) AS month_key
FROM ":SCHEMAS.WORKING:".pnl_marg_rev_marginal_method_1 m1
INNER JOIN ":SCHEMAS.CORE:".dim_location dl ON dl.location_cd = m1.location_cd
AND dl.aggr_key = m1.loc_aggr_key
INNER JOIN ":SCHEMAS.CORE:".dim_product dp ON dp.sku = m1.sku
AND dp.aggr_key = m1.sku_aggr_key
INNER JOIN ":SCHEMAS.CORE:".dim_rev_mov_type mt ON mt.mov_type_cd = m1.mov_type_cd
WHERE m1.time_key BETWEEN dl.key_start_date AND dl.key_end_date
  AND m1.time_key BETWEEN dp.key_start_date AND dp.key_end_date
  AND m1.time_key BETWEEN mt.key_start_date AND mt.key_end_date
UNION ALL
SELECT m2.endownment_time_key,m2.time_key AS negotiation_time_key,dl.location_key,m2.location_cd,m2.loc_aggr_key,dp.product_key,m2.sku,
m2.sku_aggr_key,mt.mov_type_key AS rev_type_key,m2.promotion_key AS promotion_cd,m2.int_dot_val,m2.int_curr_key AS
int_currency_key,m2.sup_dot_val,m2.sup_curr_key AS sup_currency_key,m2.system_key,m2.create_date,m2.last_updt_date,substr(m2.time_key,0,6)
AS month_key
FROM ":SCHEMAS.WORKING:".pnl_marg_rev_marginal_method_2 m2
INNER JOIN ":SCHEMAS.CORE:".dim_location dl ON dl.location_cd = m2.location_cd
AND dl.aggr_key = m2.loc_aggr_key
INNER JOIN ":SCHEMAS.CORE:".dim_product dp ON dp.sku = m2.sku
AND dp.aggr_key = m2.sku_aggr_key
INNER JOIN ":SCHEMAS.CORE:".dim_rev_mov_type mt ON mt.mov_type_cd = m2.mov_type_cd
WHERE m2.time_key BETWEEN dl.key_start_date AND dl.key_end_date
  AND m2.time_key BETWEEN dp.key_start_date AND dp.key_end_date
  AND m2.time_key BETWEEN mt.key_start_date AND mt.key_end_date
UNION ALL
SELECT m3.endownment_time_key,m3.time_key AS
negotiation_time_key,dl.location_key,m3.location_cd,m3.loc_aggr_key,dp.product_key,m3.sku,m3.sku_aggr_key,
mt.mov_type_key AS rev_type_key,m3.promotion_key AS promotion_cd,m3.int_dot_val,m3.int_curr_key AS int_currency_key,m3.sup_dot_val,
m3.sup_curr_key AS sup_currency_key,m3.system_key,m3.create_date,m3.last_updt_date, substr(m3.time_key,0,6) AS month_key
FROM ":SCHEMAS.WORKING:".pnl_marg_rev_marginal_method_3 m3
INNER JOIN ":SCHEMAS.CORE:".dim_location dl ON dl.location_cd = m3.location_cd
AND dl.aggr_key = m3.loc_aggr_key
INNER JOIN ":SCHEMAS.CORE:".dim_product dp ON dp.sku = m3.sku
AND dp.aggr_key = m3.sku_aggr_key
INNER JOIN ":SCHEMAS.CORE:".dim_rev_mov_type mt ON mt.mov_type_cd = m3.mov_type_cd
WHERE m3.time_key BETWEEN dl.key_start_date AND dl.key_end_date
  AND m3.time_key BETWEEN dp.key_start_date AND dp.key_end_date
  AND m3.time_key BETWEEN mt.key_start_date AND mt.key_end_date
UNION ALL
SELECT
rpmd.endownment_time_key,rpmd.negotiation_time_key,rpmd.location_key,rpmd.location_cd,rpmd.loc_aggr_key,rpmd.product_key,rpmd.sku,
rpmd.sku_aggr_key,rpmd.rev_type_key,rpmd.promotion_cd,rpmd.int_dot_val,rpmd.int_currency_key,rpmd.sup_dot_val,rpmd.sup_currency_key,
rpmd.system_key,rpmd.create_date,rpmd.last_updt_date,rpmd.month_key
FROM ":SCHEMAS.CORE:".REVA_MARG_D rpmd
WHERE rpmd.month_key IN (":list.REVF_LIST_OF_MONTHS:")
  AND rpmd.negotiation_time_key NOT IN (":list.REVF_LIST_OF_DAYS:");
```

ANEXO X - Estrutura da tabela do processo de receitas de investimento promocional e *script* de criação

Tabela 26 - Estrutura da tabela reva_prom_d.

Reva_prom_d	
Atributo	**Tipo de Dados**
Endownment_time_key	Decimal (8,0)
Negotiation_time_key	Decimal (38,0)
Location_key	Varchar (128)
Location_cd	Varchar (128)
Loc_aggr_key	Decimal (38,0)
Product_key	Varchar (128)
Sku	Varchar (128)
Sku_aggr_key	Decimal (38,0)
Rev_type_key	Varchar (128)
Promotion_cd	Varchar (128)
Int_dot_val	Double
Int_curr_key	Varchar (9)
Sup_dot_val	Double
Sup_curr_key	Varchar (9)
System_key	Decimal (38,0)
Create_date	Timestamp
Last_updt_date	Timestamp
Partição	
Month_key	Decimal (6,0)

Tabela 27 - *Script* de criação da tabela reva_prom_d.

***Script* de criação da tabela reva_prom_d**

```
DROP TABLE IF EXISTS DW.REVA_PROM_D;
CREATE EXTERNAL TABLE DW.REVA_PROM_D
(
ENDOWNMENT_TIME_KEY  DECIMAL(8,0)     COMMENT 'DAY OF FACT',
NEGOTIATION_TIME_KEY  DECIMAL(8,0)     COMMENT 'DAY OF FACT',
LOCATION_KEY  VARCHAR(128)     COMMENT 'LOCATION AGGREGATION KEY' ,
LOCATION_CD  VARCHAR(128)     COMMENT 'LCOATION CODE' ,
LOC_AGGR_KEY  DECIMAL(38,0)     COMMENT 'LOC AGGR KEY' ,
PRODUCT_KEY  VARCHAR(128)     COMMENT 'PRODUCT AGGREGATION KEY' ,
SKU  VARCHAR(128)     COMMENT 'SKU' ,
SKU_AGGR_KEY  DECIMAL(38,0)     COMMENT 'SKU AGGR KEY' ,
REV_TYPE_KEY  VARCHAR(128)     COMMENT 'REV AGGREGATION KEY' ,
PROMOTION_CD  VARCHAR(128)     COMMENT 'PROMOTION AGGREGATION KEY' ,
INT_DOT_VAL  DOUBLE     COMMENT 'INTERNAL VALUE',
INT_CURRENCY_KEY  VARCHAR(9)     COMMENT 'INTERNAL CURRENCY',
SUP_DOT_VAL  DOUBLE     COMMENT 'SUPPLIER VALUE',
SUP_CURRENCY_KEY  VARCHAR(9)     COMMENT 'SUPPLIER CURRENCY',
SYSTEM_KEY  DECIMAL(38,0)     COMMENT 'SYSTEM KEY',
CREATE_DATE  TIMESTAMP     COMMENT 'CREATE DATE',
LAST_UPDT_DATE  TIMESTAMP     COMMENT 'LAST UPDATE DATE'
)
PARTITIONED BY (MONTH_KEY DECIMAL(6,0) COMMENT'MONTH OF FACT AND PARTITION')
STORED AS PARQUET LOCATION '/DATA/DW/CORE/REVA_PROM_D';
```

ANEXO XI – Ficheiros *scala* dos processos *Spark*

Tabela 28 - *Script scala* do processo *spark* - S_prepare_promo_promotional_locations

***Script* scala do processo spark - S_prepare_promo_promotional_locations**
/************************************ **** DF Declarations **** ************************************/ val promotions = spark.sql("SELECT DISTINCT promotion_cd, location_cd, aggr_key FROM [SCHEMA_CORE].dim_prom_location WHERE system_cd = 'RTK' AND promotion_cd IN (SELECT DISTINCT promotion_cd FROM [SCHEMA_WORKING].pnl_prom_rev_explosion)") promotions.createTempView("PROMOTIONS") val prepare_promo_promotional_locations = spark.sql("SELECT distinct ex.promotion_cd, dpl.location_cd, dpl.aggr_key AS loc_aggr_key FROM [SCHEMA_WORKING].pnl_prom_rev_explosion ex INNER JOIN PROMOTIONS dpl ON dpl.promotion_cd = ex.promotion_cd WHERE ex.calc_type IN ('P_S_1', 'P_NH_1', 'P_S_2', 'P_NH_2', 'P_S_3', 'P_NH_3', 'P_S_4', 'P_NH_4')") prepare_promo_promotional_locations.createTempView("PREPARE_PROMO_PROMOTIONAL_LOCATIONS") /************************************ **** Consolidation Queries **** ************************************/ spark.sql("set hive.exec.dynamic.partition.mode=nonstrict") spark.sql("set mapred.*job*.queue.name=root.dsadm_financial") try((spark.sql("INSERT OVERWRITE TABLE [SCHEMA_WORKING].PNL_PROM_REV_PROMO_PROMOTIONAL_LOCATIONS SELECT * FROM PREPARE_PROMO_PROMOTIONAL_LOCATIONS")))catch{case e : Exception => println("SPARK *JOB* COMPLETED WITH ERROR")} sys.exit

Tabela 29 - *Script scala* do processo *spark* - S_prepare_promo_promotional_time_sku

***Script* scala do processo spark - S_prepare_promo_promotional_time_sku**
/********** **** DF Declarations **** **********/ val prepare_promo_promotional_time_sku = spark.sql("SELECT DISTINCT DT.TIME_KEY, FEP.PROMOTION_CD, FEP.CALC_TYPE, FEP.SKU, FEP.SKU_AGGR_KEY, FEP.PROM_START_DATE, FEP.PROM_END_DATE FROM [SCHEMA_WORKING].PNL_PROM_REV_EXPLOSION FEP, [SCHEMA_CORE].DIM_TIME DT WHERE DT.FULLDATE BETWEEN FEP.PROM_START_DATE AND FEP.PROM_END_DATE AND TIME_LEVEL = 'DAY' AND DATEDIFF(FEP.PROM_END_DATE,DT.FULLDATE) <= 45 AND FEP.CALC_TYPE IN ('P_S_1', 'P_NH_1', 'P_S_3', 'P_NH_3')") prepare_promo_promotional_time_sku.createTempView("PREPARE_PROMO_PROMOTIONAL_TIME_SKU") /********** **** Consolidation Queries **** **********/ spark.sql("set hive.exec.dynamic.partition.mode=nonstrict") spark.sql("set mapred.*job*.queue.name=root.dsadm_financial") try((spark.sql("INSERT OVERWRITE TABLE [SCHEMA_WORKING].PNL_PROM_REV_PROMO_PROMOTIONAL_TIME_SKU SELECT * FROM PREPARE_PROMO_PROMOTIONAL_TIME_SKU")))catch{case e : Exception => println("SPARK *JOB* COMPLETED WITH ERROR")} sys.exit