Manoela Camila Barbosa da Silva

Faça no seu ritmo mas não perca a hora: Tomada de decisão sob demanda do usuário utilizando dados da Web

Sorocaba, SP

07 de Agosto de 2017

Manoela Camila Barbosa da Silva

Faça no seu ritmo mas não perca a hora: Tomada de decisão sob demanda do usuário utilizando dados da Web

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC-So) da Universidade Federal de São Carlos como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação. Linha de pesquisa: *Business Intelligence* e Banco de Dados.

Universidade Federal de São Carlos – UFSCar

Centro de Ciências em Gestão e Tecnologia – CCGT

Programa de Pós-Graduação em Ciência da Computação – PPGCC-So

Orientador: Profa. Dra. Sahudy Montenegro González

Sorocaba, SP

07 de Agosto de 2017

Barbosa da Silva, Manoela Camila

Faça no seu ritmo mas não perca a hora: Tomada de decisão sob demanda do usuário utilizando dados da Web / Manoela Camila Barbosa da Silva. -- 2017.

113 f. : 30 cm.

Dissertação (mestrado)-Universidade Federal de São Carlos, campus Sorocaba, Sorocaba

Orientador: Profa. Dra. Sahudy Montenegro González

Banca examinadora: Prof. Dr. Humberto Luiz Razente, Profa. Dra. Tiemi Christine Sakata

Bibliografia

1. OLAP. 2. Self-Service BI. 3. Dados transitórios. I. Orientador. II. Universidade Federal de São Carlos. III. Título.

Ficha catalográfica elaborada pelo Programa de Geração Automática da Secretaria Geral de Informática (SIn).

DADOS FORNECIDOS PELO(A) AUTOR(A)

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências em Gestão e Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de dissertação de mestrado da candidata Manoela Camila Barbosa da Silva, realizada em 07/08/2017:

Profa. Dra. Sahudy Montenegro González
UFSCar

Prof. Dr. Humberto Luiz Razente
UFU

Profa. Dra. Tiemi Christine Sakata
UFSCar

Certifico que a sessão de defesa foi realizada com a participação à distância do membro Prof. Dr. Humberto Luiz Razente e, depois das arguições e deliberações realizadas, o participante à distância está de acordo com o conteúdo do parecer da comissão examinadora redigido no relatório de defesa da aluna Manoela Camila Barbosa da Silva.

Profa. Dra. Sahudy Montenegro González
Presidente da Comissão Examinadora
UFSCar

Ao melhor pai que eu poderia ter (in memorian).

"Mesmo sabendo que um dia a vida acaba,
a gente nunca está preparado para perder alguém."
- A última música (2010)

Agradecimentos

Agradeço,

aos meus pais, por serem minha base, meu exemplo contínuo de perseverança e por terem me ensinado o valor dos estudos. Por me darem sempre mais do que tinham e me apoiarem ao longo de todo o caminho.

À bulldog da minha irmã, Mikaele, pela parceria que dura uma vida, apesar da sua atual fase rebelde.

Ao Heitor, meu pequeno príncipe, por trazer tanta alegria e fofura para a minha vida.

Aos amigos Bia e Cossa, pelas horas empregadas em classificação quando poderiam estar dormindo e pelas outras tantas como psicólogos pessoais. Mas, principalmente, pela camaradagem e momentos de descontração ao longo do percurso, porque no fim é o que mais vale.

Ao meu amigo Breno, por estar sempre disposto a compartilhar seus conhecimentos e me auxiliar quando necessário, pela paciência constante e pelas (muitas) horas gastas discutindo métodos.

Ao melhor presente que o intercâmbio me deu, Michel, por ser esse amigo que mesmo distante é tão presente.

À little Ari e a Bru, por serem tão boas amigas e por terem sido meu apoio quando eu mais precisei de um, palavras jamais serão o suficiente para agradecê-las.

A todos os amigos com os quais eu recusei mais compromissos do que era possível e continuaram meus amigos. Obrigada pela compreensão e por não desistirem de mim.

Por último, à minha orientadora, Sahudy, pelos ensinamentos e dedicação constantes, não somente na academia, mas na vida. Por ser não somente orientadora, mas amiga. Por escutar minhas alegrias e frustrações, fossem elas científicas ou pessoais, e por compartilhar as suas comigo também. Pelos bretes, pelos conselhos e pelo carinho, ainda que disfarçado. São coisas que não têm nenhum valor científico, mas cuja valia é imensurável.

"Aqueles que passam por nós, não vão sós, não nos deixam sós.
Deixam um pouco de si, levam um pouco de nós".
(Antoine de Saint-Exupéry)

Resumo

Na atual era do conhecimento, com o crescimento contínuo do volume de dados da Web e onde decisões de negócio devem ser feitas de maneira rápida, os mecanismos tradicionais de *BI* se tornam cada vez menos precisos no auxílio à tomada de decisão. Em resposta a este cenário surge o conceito de *BI* 2.0, que se trata de um conceito recente e se baseia principalmente na evolução da Web, tendo como uma das principais características a utilização de fontes Web na tomada de decisão. Porém, dados provenientes da Web tendem a ser voláteis para serem armazenados no DW, tornando-se uma boa opção para dados transitórios. Os dados transitórios são úteis para consultas de tomada de decisão em um determinado momento e cenário e podem ser descartados após a análise. Muitos trabalhos têm sido desenvolvidos em relação à *BI* 2.0, mas ainda existem muitos pontos a serem explorados. Este trabalho propõe uma arquitetura genérica para SSDs, que visa integrar dados transitórios, provenientes da Web, às consultas de usuários no momento em que o mesmo necessita deles para a tomada de decisão. Sua principal contribuição é a proposta de um novo operador *OLAP*, denominado Drill-Conformed, capaz de realizar a integração dos dados de maneira automática e fazendo uso somente do domínio de valores dos dados transitórios. Além disso, o operador tem o intuito de colaborar com a Web semântica, a partir da disponibilização das informações por ele descobertas acerca do domínio de dados utilizado. O estudo de caso é um sistema de disponibilização de *streamings*. Os resultados dos experimentos são apresentados e discutidos, mostrando que é possível realizar a integração dos dados de maneira satisfatória e com bons tempos de processamento para o cenário aplicado.

Palavras-chaves: OLAP. Integração de dados. Web semântica. Self-Service BI. Dados transitórios.

Abstract

In the current knowledge age, with the continuous growth of the web data volume and where business decisions must be made quickly, traditional BI mechanisms become increasingly inaccurate in order to help the decision-making process. In response to this scenario rises the BI 2.0 concept, which is a recent one and is mainly based on the Web evolution, having as one of the main characteristics the use of Web sources in decision-making. However, data from Web tend to be volatile to be stored in the DW, making them a good option for situational data. Situational data are useful for decision-making queries at a particular time and situation, and can be discarded after analysis. Many researches have been developed regarding to BI 2.0, but there are still many points to be explored. This work proposes a generic architecture for Decision Support Systems that aims to integrate situational data from Web to user queries at the right time; this is, when the user needs them for decision making. Its main contribution is the proposal of a new OLAP operator, called Drill-Conformed, enabling data integration in an automatic way and using only the domain of values from the situational data.In addition, the operator collaborates with the Semantic Web, by making available the semantics-related discoveries. The case study is a streamings provision system. The results of the experiments are presented and discussed, showing that is possible to make the data integration in a satisfactory manner and with good processing times for the applied scenario.

Key-words: OLAP. Data integration. Semantic Web. Self-Service BI. Situational data.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

3FN	Terceira Forma Normal
ACID	Atomicity, Consistency, Isolation, Durability (Atomicidade, Consistência, Isolamento e Durabilidade)
API	Application Programming Interface (Interface de Programação de Aplicativos)
BI	Business Intelligence
DeTEC	Detecção e extração, Transformação, Enriquecimento e Carga
DW	Data Warehouse
ER	Entidade-Relacionamento
ETL	Extract Transform Load (Extração, Transformação e Carga)
GUI	Graphical User Interface (Interface gráfica do utilizador)
IMDb	Internet Movie Database
JSON	JavaScript Object Notation (Notação de Objetos JavaScript)
MER	Modelo Entidade-Relacionamento
NLP	Natural Language Processing (Processamento de Linguagem Natural)
OLAP	On-line Analytical Processing (Processamento Analítico On-line)
OLTP	On-line Transaction Processing (Processamento de transações em tempo real)
RDF	Resource Description Framework
REST	Representation State Transfer (Tranferência de Estado Representacional)
SaaS	Software as Service (Software como Serviço)
SQL	Structured Query Language (Linguagem de Consulta Estruturada)
SSD	Sistema de Suporte à Decisão
TI	Tecnologia da Informação

URL	Uniform Resource Locator (Localizador Uniforme de Recursos)
XML	eXtensible Markup Language

Sumário

1 Introdução

Vive-se atualmente a era do conhecimento e inclusão digital, onde informações surgem e se transformam muito dinamicamente e decisões de negócio devem ser feitas de maneira rápida, precisa e utilizando-se do maior conhecimento e visões do domínio possíveis. Tal cenário tem causado a necessidade de evolução dos mecanismos de *Business Intelligence* (*BI*) tradicionais. Essa evolução diz respeito, principalmente, à obtenção de dados em tempo real e à utilização da riqueza e diversidade de informações provenientes da Web e suas mídias sociais como fonte de dados (ABELLÓ et al., 2015) (MANSMANN et al., 2014). Ambos os tópicos são objetos de estudo deste trabalho.

Os Sistemas de Suporte à Decisão (SSDs) são sistemas que auxiliam o processo de tomada de decisão dentro de organizações. O componente chave desses sistemas são os *Data Warehouses* (*DWs*), que consistem de um banco de dados central, orientado a assunto, que armazena, de maneira padronizada e consistente, dados provenientes de diversas e heterogêneas fontes e sobre o qual todo o processamento é realizado.

Data Warehouses tradicionais geralmente são alimentados periodicamente (semanalmente, por exemplo), sendo essa periodicidade definida pelo gestor do SSD, de acordo com a necessidade do domínio de negócio sobre o qual o mesmo será aplicado. Esse intervalo de tempo, que ocorre entre uma alimentação e outra, ocasiona em um atraso entre a geração da informação na sua fonte e a disponibilização da mesma no processo de tomada à decisão. Com o crescimento explosivo das novas informações sendo disponibilizadas a cada minuto nos últimos anos, esse atraso passou a ocasionar em uma perda de dados significativa, dependendo do cenário, aumentando cada vez mais a demanda pela utilização de dados em tempo real, ou o mais próximo possível disso (VAISMAN; ZIMÁNYI, 2012).

Com o crescimento da *World Wide Web* e o advento da Web 2.0, a Internet tornou-se um ambiente mais dinâmico, onde o próprio usuário é criador de conteúdo, tornando-a uma fonte rica de informação para a tomada de decisão. Além disso, expandiu-se a heterogeneidade das estruturas e formatos dos dados disponíveis para análise, dentre eles, textos, imagens e vídeos. Dessa forma, aumentou-se a necessidade de mecanismos de suporte à decisão capazes de armazenar e manipular dados semi ou não estruturados. Dados semi estruturados são aqueles que não possuem estrutura definida mas incluem informações sobre seu esquema, ainda que na forma de metadados, enquanto que os não estruturados são caracterizados por não possuírem qualquer estrutura ou informação a despeito de seu esquema (IGLESIAS et al., 2015).

Uma das principais características dos SSDs tradicionais é que eles armazenam os dados extraídos das fontes de maneira permanente no *DW* (o que chamaremos neste

trabalho de dados estacionários), a fim de exibir informações de maneira histórica. Porém, como exposto por Etcheverry e Vaisman (2012), dados provenientes de fontes externas, como mídias sociais, tendem a ser muito voláteis para serem armazenados permanentemente no *DW*. Com base nisso, a utilização dos chamados dados transitórios para esse tipo de dados se tornou uma boa opção. Dados transitórios consistem em dados que são úteis para a tomada de decisão em um determinado momento e cenário, mas após isso podem ser descartados, pois não tem utilidade para as consultas fora deste contexto.

Todo esse novo cenário foi responsável pelo surgimento da *BI* 2.0 (CHEN; CHIANG; STOREY, 2012). Dentre os diversos aspectos propostos por tal conceito encontram-se a utilização do volume de dados disponibilizados pela Web e suas mídias sociais, a utilização de dados transitórios e a obtenção de dados em tempo (próximo do) real (TRUJILLO; MATÉ, 2012). Tais características visam tirar proveito da grande diversidade de opiniões e *feedbacks* oferecidos pela Web 2.0 e manter uma visão atualizada das fontes de dados, tornando a tomada de decisão mais rica e precisa.

Dentre as abordagens que vêm sendo propostas para obtenção de dados em tempo real se encontram os *Rigth-Time Data Warehouses*, que se baseiam no princípio de que os dados devem ser buscados nas fontes e atualizados no *DW* conforme forem necessários (THOMSEN; PEDERSEN; LEHNER, 2008). Desta maneira, o *DW* passa a refletir uma imagem das fontes de dados do momento em que o usuário realizou a consulta, e não uma imagem de dias atrás.

Outro componente importante dos SSDs são os servidores *OLAP* (*Online Analytical Processing*), que são responsáveis pelo processamento das consultas realizadas sobre o *Data Warehouse*, juntamente com alguns operados de navegação disponibilizados, denominados operadores *OLAP*. Esses últimos são responsáveis por permitir a navegação do usuário através dos dados, fornecendo diversas visões do cenário. Porém, como exposto por González e Berbel (2014), os operadores *OLAP* tradicionais também não foram projetados para processar dados não estruturados, exigindo adaptações no processo e/ou criação de novos operadores quando nessa nova geração de *BI*.

Dado os fatos apresentados, a presente dissertação propõe uma arquitetura para Sistemas de Suporte à Decisão genérica, que visa atender os aspectos da *BI* 2.0, através da integração e combinação de dados transitórios e estacionários em tempo real, visando a autonomia do usuário durante o processo de tomada de decisão. O componente principal da arquitetura é um novo operador *OLAP*, denominado Drill-Conformed, que tem como intuito ser um operador genérico, aplicável a diversos cenários. O operador visa realizar a integração dos dados estacionários com transitórios de maneira automática, sem a necessidade de interferência de um especialista em *BI*, e fazendo uso somente do domínio de valores dos dados Web obtidos.

1.1 Objetivos

O objetivo deste trabalho é propor uma arquitetura genérica que atenda as principais características propostas pela nova geração de *BI*. Essa arquitetura visa realizar a integração de dados estacionários com transitórios, provenientes da Web e suas mídias sociais, em tempo próximo do real e de maneira automática, provendo autonomia ao usuário. Dessa forma, não se faz necessária a intervenção de um especialista em *BI* e/ou *TI* para realização das consultas.

Para integração dos dados é proposto um novo operador *OLAP*, denominado Drill-Conformed, que é o ponto principal da proposta apresentada. O objetivo do operador é realizar o cruzamento dos dados transitórios e estacionários de maneira autônoma, utilizando somente o domínio de valores dos atributos contidos nos mesmos. A intenção deste é ser capaz de realizar tal junção quando não há informações semânticas prévias, como metadados, acerca dos atributos presentes nos dados. Além disso, ele visa permitir que usuários em todos os níveis de conhecimento técnico sejam capazes de realizar consultas que englobem os dois tipos de dados por si próprios.

Definidos os objetivos deste trabalho, as principais questões que visasse ser possível responder ao fim desta dissertação são expostas da seguinte forma:

1. A obtenção e junção dos dados pode ser feita de maneira automática e satisfatória, utilizando apenas o conjunto de valores de seus domínios e sem conhecimento prévio acerca de sua semântica?

2. É possível realizar o processo de obtenção, tratamento e integração dos dados Web em tempo considerado real, ou o mais próximo disso possível?

1.2 Organização do trabalho

Com a finalidade de facilitar a compreensão por parte do leitor, o presente texto foi estruturado da seguinte forma:

- O Capítulo 2 apresenta uma visão geral dos conceitos de Sistemas de Suporte à Decisão, *Data Warehouse*, modelo multidimensional, processo de *ETL*, servidores *OLAP* e conceitos relacionados à *BI* 2.0. Além disso, a revisão da literatura de pesquisas recentes, relacionadas ao tema deste trabalho, são expostas.

- O Capítulo 3 descreve a arquitetura e operador propostos.

- O Capítulo 4 exibe o protótipo Web desenvolvido para verificar o funcionamento e a interação com o operador Drill-Conformed.

- O Capítulo 5 expõe os resultados obtidos por meio da avaliação experimental realizada sobre os componentes.
- Por fim, as conclusões e trabalhos futuros são apresentados no Capítulo 6.

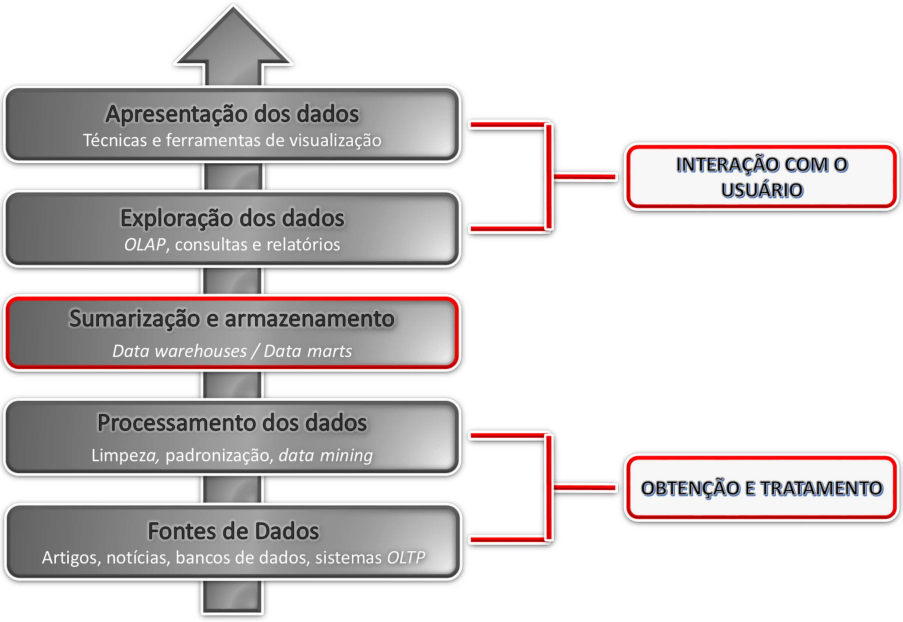
2 Fundamentos e Estado da Arte

Neste capítulo são apresentados os fundamentos teóricos acerca de: Sistemas de Suporte à Decisão, *Data Warehouse*, modelo multidimensional, *ETL*, servidores *OLAP* e conceitos relacionados à nova geração de *BI*. Posteriormente, são apresentados alguns trabalhos e pesquisas recentes, relacionados com a evolução dos mecanismos de *BI*, a fim de que os mesmos sejam capazes de lidar com a alta dinamicidade e heterogeneidade das informações do cenário atual, tornando-os mais ricos e flexíveis.

2.1 Fundamentos sobre Sistema de Suporte à Decisão

Em *Business Intelligence*, Sistemas de Suporte à Decisão são um conjunto de sistemas, ferramentas e tecnologias que, juntos, auxiliam no processo de tomada de decisão dentro de uma instituição. De maneira geral, os SSDs são responsáveis por definir uma arquitetura que determine um fluxo de informações e técnicas que favoreçam e auxiliem a tomada de decisão. Eles recuperam e armazenam informações de diversas fontes, apresentando-as posteriormente de forma unificada aos usuários. A Figura 1 apresenta, de maneira genérica, o fluxo de informações de um SSD.

Figura 1: Arquitetura genérica de um Sistema de Suporte a Decisão



Fonte: Produzido pela autora

Tais sistemas podem ser subdivididos em três etapas básicas: i) obtenção e tra-

tamento dos dados de suas fontes; ii) sumarização e armazenamento dos mesmos; e iii) interação com o usuário. Estas etapas podem ser representadas, respectivamente, pelo processo de *ETL*, *Data Warehouse* e Processamento Analítico On-line.

2.1.1 Processo de Extração, Transformação e Carga

O processo de *ETL - (E)xtract, (T)ransform, (L)oad* é uma etapa fundamental dos SSDs, através do qual ocorre toda a transição dos dados de suas fontes para a ferramenta de auxílio à tomada de decisão (KIMBALL; ROSS, 2013). Ele é responsável pela extração dos dados das fontes, limpeza e transformação dos mesmos. É através deste processo que os dados, retirados de múltiplas, heterogêneas e distribuídas bases, são padronizados, sumarizados e integrados de maneira consistente ao *Data Warehouse*.

Para que o processo de *ETL* seja executado, é necessário que as fontes de dados, das quais as informações serão obtidas, tenham sido previamente definidas pelo gestor do sistema. Em mecanismos de *BI* tradicionais, essas fontes são, em sua maioria, bases de dados provenientes de sistemas transacionais, conhecidos como *Online Transaction Processing (OLTP)*. Kimball e Ross (2013) citam alguns critérios para criação de um bom processo de *ETL*, dentre os quais se encontram dois que são importantes considerar na escolha das fontes de dados:

- **Necessidades do negócio:** referente às questões que o sistema visa responder no intuito de auxiliar a tomada de decisão. As fontes de dados devem prover informações úteis e relacionadas ao cenário sob o qual a tomada de decisão será realizada.

- **Qualidade dos dados:** relacionado com a consistência e completude dos dados provenientes das fontes selecionadas. Fontes de dados com uma quantidade significativa de dados faltantes e/ou anomalias tendem a não dar uma visão correta do cenário, além de exigir uma complexidade maior da etapa de transformação dos dados.

Uma vez que as fontes de dados foram definidas, o processo de *ETL* pode então ser executado. As fases desse processo podem ser melhor descritas como (JENSEN; PEDERSEN; THOMSEN, 2010):

- **Extração:** é a primeira etapa não somente do processo de *ETL*, mas do sistema como um todo. Tal etapa consiste em ler e entender os dados das fontes previamente definidas, de forma a extrair somente aqueles que são úteis para o processo de decisão ao qual serão aplicados. Além disso, essa etapa deve ser projetada de forma a extrair somente os dados adicionados ou atualizados desde a última extração, de forma a diminuir o grande volume de dados.

- **Transformação:** essa é a fase responsável por transformar os dados obtidos das diversas fontes compatíveis com o esquema de *Data Warehouse* definido, de forma que eles possam ser armazenados no mesmo de maneira consistente. Essa transformação pode envolver vários processos, desde a integração e padronização dos dados das muitas fontes até a inserção de valores faltantes, dependendo do modelo de dados e qualidade dos dados obtidos. As transformações mais comumente aplicadas são: limpeza, remoção de conflitos, adição de campos faltantes, normalização e conversões de formatos. Também é nessa fase que técnicas de mineração de dados (*data mining*) são aplicadas, a fim de enriquecer a informação obtida. Além disso, no intuito de manter a integridade referencial, essa fase também é responsável pela sumarização dos dados e geração das chaves artificiais.

- **Carga:** esse último passo é o responsável por alimentar o *Data Warehouse* com os novos dados. A atualização costuma lidar com grandes volumes de dados e, desta forma, esse tende a ser um processo demorado, durante o qual, normalmente, o *Data Warehouse* fica *offline*. No intuito de agilizar esse processo, algumas técnicas são comumente aplicadas, como, por exemplo, remoção dos índices do *DW* durante a atualização e/ou paralelização do processo.

O processo de *ETL* costuma ser o gargalo de um Sistema de Suporte à Decisão. Além do tempo requerido para entender as fontes, o que seus dados significam, quando e como extrair esses dados e como essas diversas fontes podem ser combinadas, a extração e manipulação do grande volume de dados com que esse processo lida também requerem um tempo de processamento considerável.

2.1.2 *Data Warehouse*

O *Data Warehouse* é o componente central de um SSD. Consiste de um banco de dados centralizado, implementado separadamente dos bancos de dados operacionais da instituição, devido principalmente à modelagem e uso de mecanismos especiais (CHAUDHURI; DAYAL, 1997), que provê um modelo de armazenamento histórico de dados, armazena informações provenientes de diversas fontes, internas e externas, e favorece análises multidimensionais sob um grande volume de dados (HAN; KAMBER; PEI, 2012).

É definido por Inmon (2002), um dos nomes mais conhecidos na construção do *Data Warehouse*, como:

- **Orientada a assunto:** diferentemente das bases de dados operacionais existentes dentro das instituições, que armazenam dados acerca de transações individuais realizadas, um *DW* é organizado em torno de assuntos cuja análise é importante

para a instituição como, por exemplo, produtos, clientes etc, de forma a simplificar a exibição dos mesmos ao usuário final.

- **Integrado:** projetado para armazenar dados de diversas e heterogêneas fontes de maneira sincronizada e obedecendo a um padrão pré-definido. A etapa de transformação do processo de *ETL* é responsável por garantir a padronização desses dados.

- **Variável com o tempo:** *Data Warehouses* armazenam dados de maneira histórica, a fim de permitir a análise de um determinado assunto durante determinados períodos, dando uma perspectiva da situação sendo analisada com o passar do tempo.

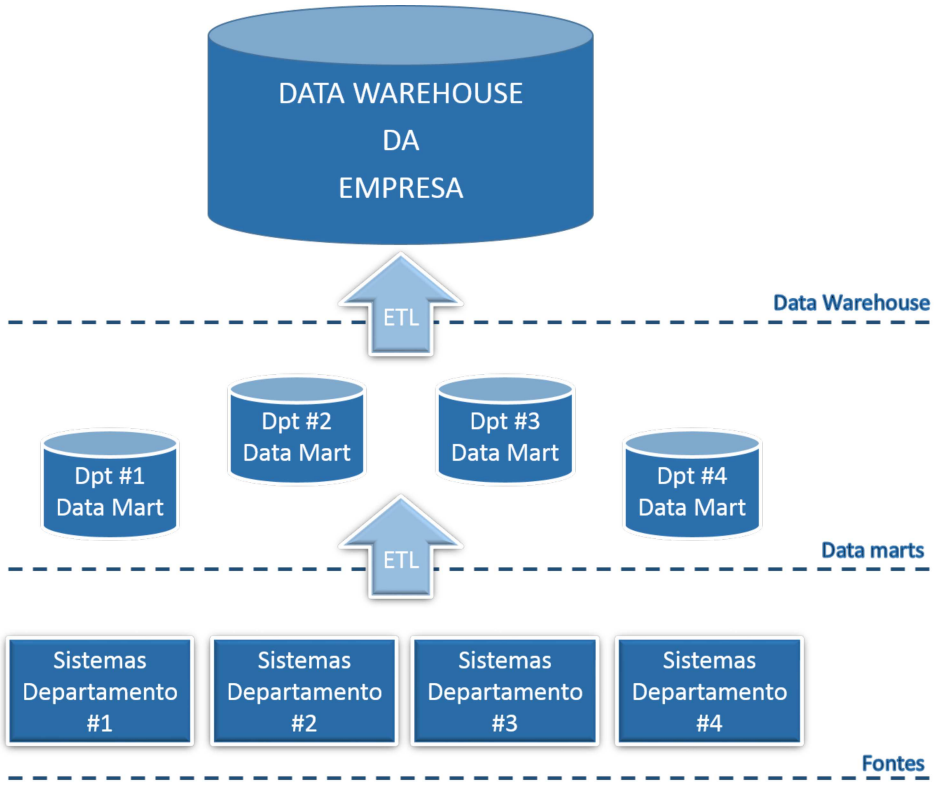
- **Não volátil:** diferentemente das bases de dados transacionais, cuja informação muda com o decorrer do tempo, *DWs* dificilmente sofrem modificações nos dados já inseridos, salvo os casos em que são necessárias correções. Uma vez que o intuito é permitir uma visualização de dados histórica, os dados armazenados são mantidos permanentemente, sendo realizadas apenas operações de inserção e busca.

Nos Sistemas de Suporte à Decisão tradicionais, *DWs* são alimentados periodicamente (semanalmente ou mensalmente, por exemplo), sendo tal periodicidade definida pelo gestor do sistema, de acordo com a necessidade do domínio de negócio a ser gerenciado. Esse intervalo de tempo, que ocorre entre uma alimentação e outra, ocasiona em um atraso entre a geração de uma informação na sua base fonte e sua disponibilização no processo de tomada à decisão. Dessa maneira, tais *DWs* tendem a representar dados históricos, com a visão mais atual sendo um momento no passado (JÖRG; DESSLOCH, 2010), representando a situação das fontes de dados no momento de sua última atualização, em vez de uma visão das mesmas no momento atual.

Uma outra característica desses sistemas tradicionais é que eles proveem apenas dados estacionários para a tomada de decisão. Da mesma forma que Abelló et al. (2013), durante este trabalho denominaremos dados estacionários aqueles dados que, antes ou após a sua utilização no processo de análise, são armazenados de forma permanente do *DW*, tornando-se parte constante da tomada de decisão e sendo processados em todas as análises.

Existem, também, os *data marts*, que são um subconjunto do *Data Warehouse*. Enquanto um *DW* centraliza informações acerca dos diversos setores e assuntos dentro de uma organização, um *data mart* armazena informações somente de um determinado setor do negócio e sobre um único assunto, sem a necessidade de cruzamento de informações com os demais departamentos (JENSEN; PEDERSEN; THOMSEN, 2010). Dessa forma, o *DW* de uma instituição pode ser construído a partir da junção de dois ou mais de seus *data marts*.

Figura 2: *Data Marts* e *Data Warehouse*



Fonte: Produzido pela autora

A Figura 2 apresenta uma representação gráfica do fluxo de informações em uma instituição que possui *data marts* e cujo *Data Warehouse* é constituído unicamente da junção destes. Vale ressaltar que, a junção de *data marts* em um único *DW* é apenas uma das formas de se construir um *DW* dentro de uma instituição.

2.1.2.1 Modelagem Multidimensional

A modelagem de um *Data Warehouse* difere do tradicional Modelo Entidade Relacionamento (MER) largamente utilizado em sistemas *OLTP*. Estes últimos prezam pelo alto nível de normalização de suas bases de dados, uma vez que fornecem eficiência no processamento *on-line* de transações, principalmente *inserts* e *updates*. *Data Warehouses*, por sua vez, requerem um modelo de dados orientado a assunto, que facilite a tomada de decisão e que tenha alta performance na recuperação dos dados (HAN; KAMBER; PEI, 2012) (KIMBALL; ROSS, 2013).

O modelo mais comumente utilizado para a modelagem de *DWs* é o multidimensional. Tal modelo é largamente utilizado nesse cenário devido ao fato de fornecer informação de maneira entendível ao usuário final, permitir a fácil visualização dos dados de diversas perspectivas e à alta performance em consultas, causada pela simplicidade do modelo e,

em parte, a não normalização dos dados (KIMBALL; ROSS, 2013).

Embora use uma modelagem diferente, os *Data Warehouses* também podem ser implementados utilizando um sistema de banco de dados relacional, divergindo somente na forma como o mesmo é modelado. Tal modelagem apresenta uma representação bidimensional de cubos multidimensionais e consiste, essencialmente, de dois tipos de tabelas: tabela fato e tabela dimensão.

A tabela fato é uma tabela central e normalizada, que representa uma relação de multiplicidade entre as dimensões. Ela é responsável pelo armazenamento do objeto de análise, que consiste de medidas de negócio sumarizadas através de níveis de detalhes representados pelas dimensões. Tais medidas são dados quantitativos e, geralmente, consistem de valores numéricos, sumarizados utilizando operações matemáticas, como soma e média.

A tabela dimensão, por outro lado, é composta de dados qualitativos, responsáveis pela representação contextual relacionada às medidas da tabela fato. É através dessas tabelas que o usuário é capaz de navegar pelos dados e analisar o objeto de análise de diversas perspectivas. Existem algumas características importantes em relação a esse tipo de tabela:

- **Chaves artificiais**: tais tabelas são caracterizadas pelo uso de chaves primárias artificiais incrementais, com o intuito de minimizar a complexidade dos relacionamentos, diminuir o tamanho da tabela fato, maximizar o desempenho das buscas e separar as chaves primárias do *DW* daquelas pertencentes às fontes de dados.

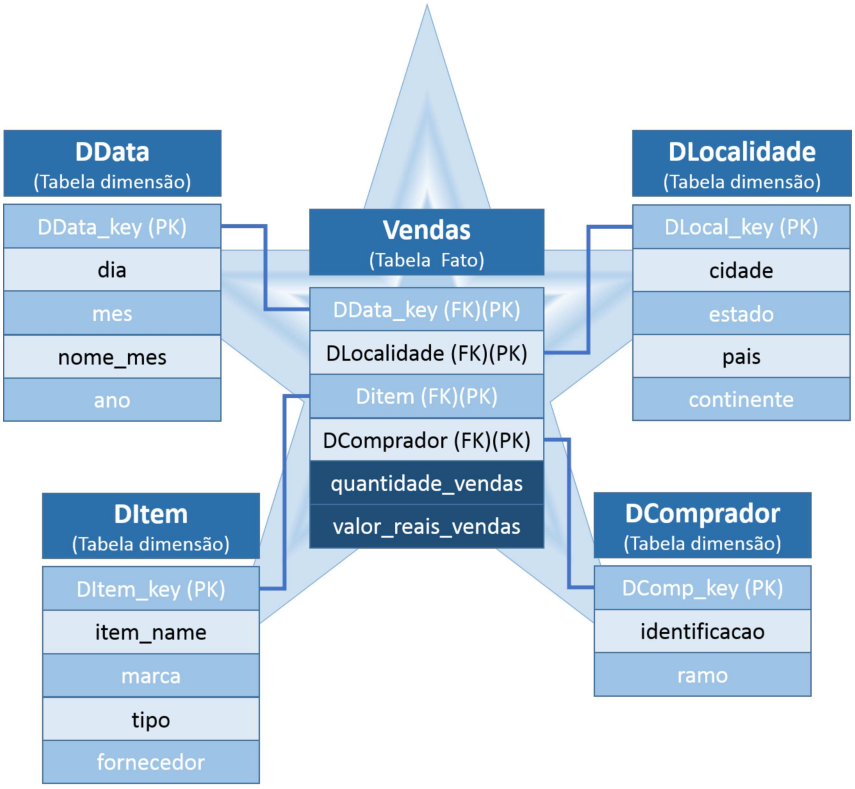
- **Hierarquias**: tabelas dimensões são compostas de níveis de hierarquias. Uma hierarquia, como definido por Han, Kamber e Pei (2012), é uma sequência de mapeamentos de conceitos mais específicos para conceitos mais gerais. Por exemplo, em uma dimensão `Data`, contendo os campos `dia`, `mês` e `ano`, dia seria o conceito mais específico e, portanto, nível mais baixo da hierarquia, enquanto ano seria o conceito mais geral e de mais alto nível, passando por mês no mapeamento.

- **Dimensões básicas**: existem 4 (quatro) dimensões básicas que geralmente estão presentes em um *Data Warehouse*, responsáveis por responder às questões base: quem, quando, onde e o que.

A modelagem multidimensional possui três tipos de modelos: estrela (*star*), floco de neve (*snowflake*) e constelação (*fact constellation*). Sendo o esquema em estrela o modelo base e os demais uma variação deste primeiro.

O modelo estrela, como mostra a Figura 3, é composto de uma única tabela fato, contendo todas as métricas, e um conjunto de tabelas dimensões que contém toda a

Figura 3: Modelo Estrela



Fonte: Derivada daquelas apresentadas por Han, Kamber e Pei (2012) e Kimball e Ross (2013)

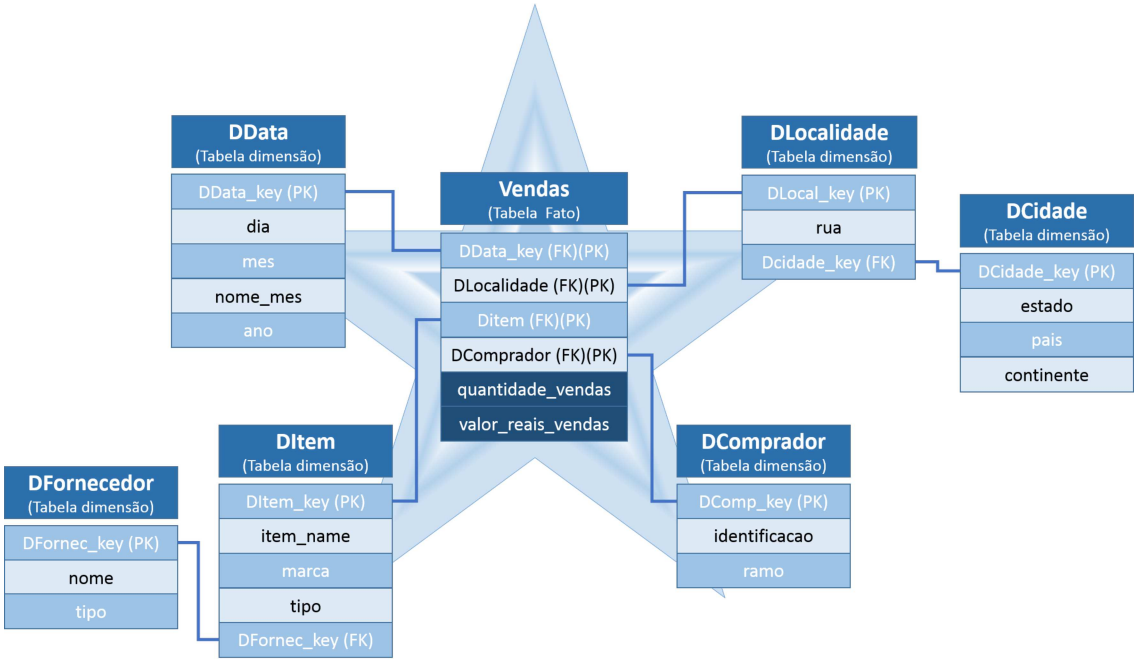
informação qualitativa de maneira não normalizada. A tabela fato é relacionada, através de uma relação muitos-para-um e das chaves artificiais, com todas as tabelas dimensões, mantendo assim a integridade referencial.

O modelo floco de neve, por sua vez, é uma variação do modelo estrela, onde algumas ou todas as dimensões passam a estar normalizadas na 3FN (Terceira Forma Normal), dividindo os dados de uma dimensão em mais de uma tabela. Dessa forma, além de estarem ligadas a tabela fato, as tabelas dimensões podem agora também estarem ligadas a outras tabelas dimensões. Este modelo é apresentado na Figura 4.

Por fim, temos o modelo constelação, que consiste em um modelo contendo múltiplas tabelas fato que compartilham entre si tabelas dimensões. Tal modelo, apresentado na Figula 5, pode ser entendido como uma junção de modelos estrelas, daí o nome constelação (HAN; KAMBER; PEI, 2012).

Existem, ainda, alguns tipos especiais de dimensões que podem ser utilizadas na modelagem de um *Data Warehouse*, dentre as quais se encontram (JENSEN; PEDERSEN; THOMSEN, 2010):

Figura 4: Modelo Floco de Neve



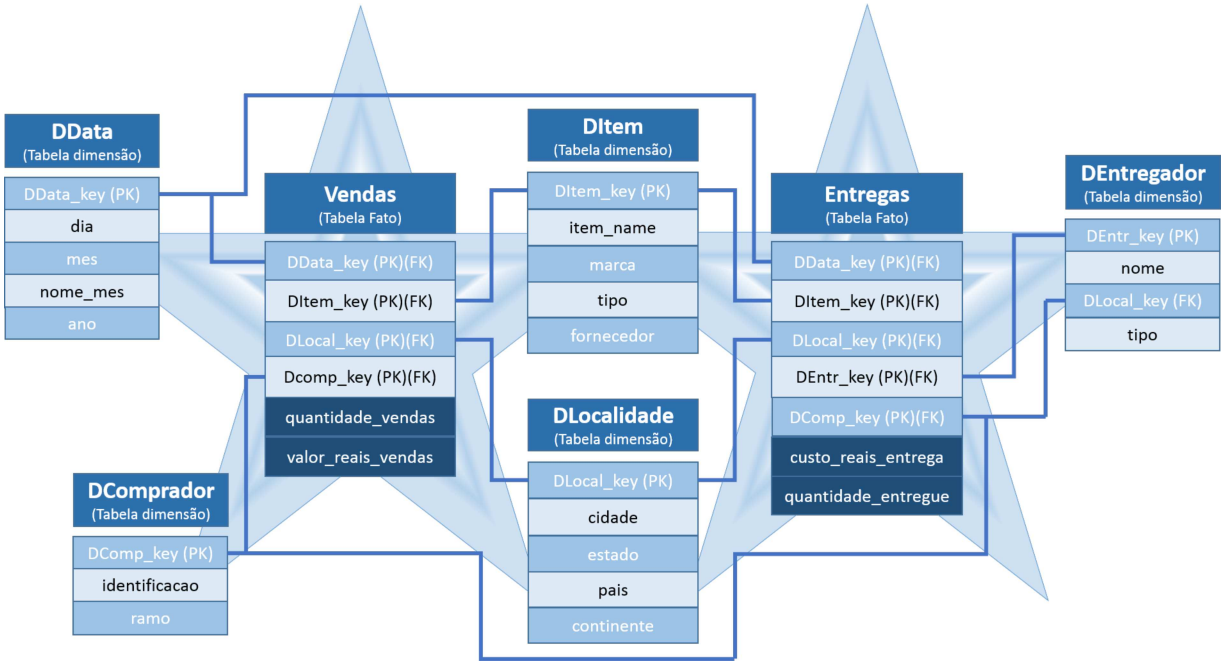
Fonte: Derivada daquelas apresentadas por Han, Kamber e Pei (2012) e Kimball e Ross (2013)

- **Minidimensões**: consiste em dividir uma dimensão muito grande, em quantidade de atributos, em dimensões menores, onde todas as novas dimensões são conectadas a tabela fato. Tais dimensões são frequentemente utilizadas para controlar o crescimento explosivo de dimensões cuja parte ou todos os atributos são continuamente mutáveis. Geralmente divide-se a dimensão de forma que os atributos que têm frequência similar de modificações fiquem na mesma minidimensão.

- **Outriggers**: são dimensões obtidas da divisão de dimensões grande e que são referenciadas, por meio de chave estrangeira, por outras dimensões. Geralmente utilizadas para separar atributos de baixa cardinalidade dos demais, de forma a diminuir o espaço utilizado e melhorar a performance de atualização dos dados. Vale ressaltar que, apesar das dimensões estarem relacionadas através de uma chave estrangeira, *outriggers* diferem do modelo floco de neve, uma vez que as tabelas não precisam estar normalizadas.

- **Dimensões degeneradas**: é uma dimensão que contém um único campo, geralmente um identificador único, e nenhum atributo descritivo. Em representações relacionais, essa dimensão é frequentemente representada sem tabela associada, com o identificador inserido diretamente na tabela fato. Tal abordagem é utilizada com o intuito de diminuir o espaço utilizado no banco e operações de *join* durante uma consulta.

Figura 5: Modelo Constelação



Fonte: Derivada daquelas apresentadas por Han, Kamber e Pei (2012) e Kimball e Ross (2013)

Existe também um tipo especial de tabela fato, conhecido como tabela fato sem fatos (*factless fact table*). Esse tipo de tabela contém apenas as combinações de chaves estrangeiras entre as dimensões e nenhuma medida. É utilizada para obter informações acerca dos relacionamentos entre as dimensões, independentemente de um evento ter ou não ocorrido.

2.1.3 Processamento Analítico On-line

De maneira diferente de sistemas *OLTP*, que realizam processamento transacional, sendo responsáveis pelo armazenamento das transações de operações diárias, em sua maioria atômicas, dentro de uma organização, *Data Warehouses* são projetados para processamento analítico. Tal processamento é realizado através de sistemas *OLAP*, de grandes volumes de dados, de forma a facilitar a tomada de decisão. Han, Kamber e Pei (2012) apresentam as principais diferenças entre os dois tipos de sistemas:

- **Usuários e orientação do sistema:** sistemas *OLTP* são orientados a clientes, isto é, são sistemas desenvolvidos para satisfazer uma necessidade específica do negócio do cliente, como por exemplo, gerenciamento sobre as vendas da empresa, e geralmente são utilizados pelos empregados de mais baixo nível. Sistemas *OLAP*, por outro lado, são orientados a assunto e desenvolvidos para análise de grandes volumes de dados,

sendo utilizados por profissionais de mais alto nível na empresa, como, por exemplo, os executivos, que possuem conhecimento do negócio e geralmente são responsáveis por tomadas de decisões.

- **Conteúdo**: enquanto sistemas *OLTP* gerenciam dados atuais, representando as transações diárias do negócio, sistemas *OLAP* lidam com grandes volumes de dados históricos, de maneira sumarizada e em vários níveis de detalhe, facilitando assim o uso desses dados na tomada de decisão.

- **Modelagem do banco de dados:** como citado na Seção 2.1.2.1, sistemas *OLTP* geralmente utilizam o modelo Entidade Relacionamento (ER) tradicional, enquanto que sistemas *OLAP* utilizam a modelagem multidimensional.

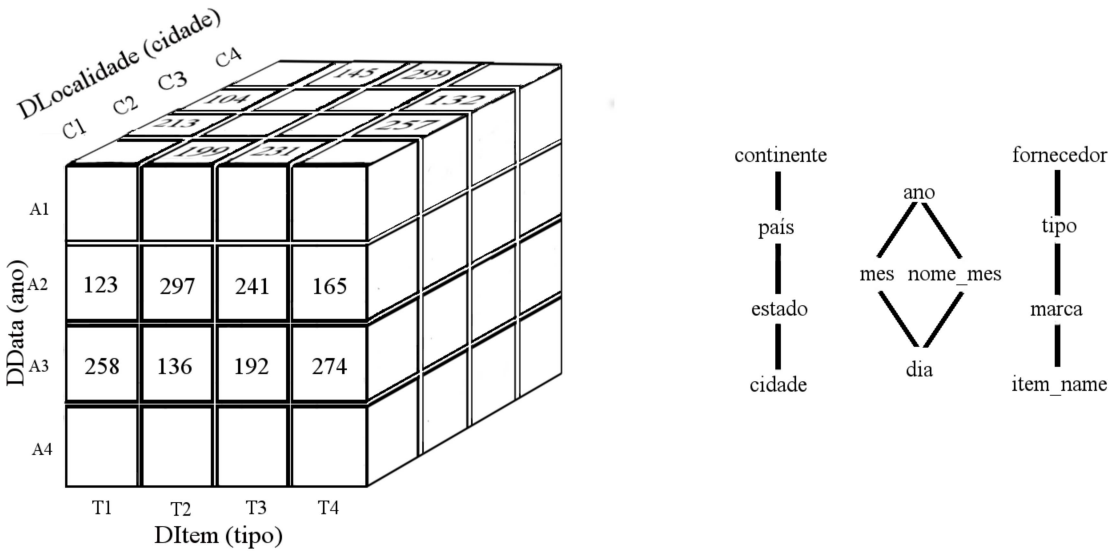
- **Tipos de acesso**: devido ao tipo de dado e orientação dos sistemas *OLTP*, o acesso aos seus dados consiste de transações atômicas e requerem controle de concorrência, devido ao grande número de operações de inserção e atualização. Os sistemas *OLAP*, por sua vez, realizam, em sua grande maioria, operações de leitura apenas, uma vez que os dados armazenados tendem a não ser modificados, salvo exceções.

O servidor *OLAP* é um componente importante dos Sistemas de Suporte à Decisão, cuja função é a de acessar o *DW* e exibir seus dados na forma de um cubo multidimensional, também conhecido como cubo *OLAP*, para a camada de interação com o usuário. Cada célula desse cubo representa uma medida, ou conjunto de medidas, representando os fatos e a contextualização dos mesmos. Apesar de a palavra cubo remeter a uma figura geométrica tridimensional, um cubo *OLAP* é `n`-dimensional, onde `n` é o número de dimensões do *Data Warehouse*.

A Figura 6 apresenta uma representação gráfica de um cubo *OLAP*, baseado nas dimensões utilizadas na Figura 3. As dimensões contidas no cubo são `DData`, `DLocalidade` e `DITem`, de forma que `DLocalidade` está agregada por `cidade` (`C1`,`C2`, `C3`, `C4`), `DData` por `ano` (`A1`, `A2`, `A3`, `A4`) e `DItem` por `tipo` (`T1`, `T2`, `T3`, `T4`). Do lado direito da figura são apresentadas as hierarquias presentes no cubo, sendo o nível mais baixo aquele contido na parte inferior das mesmas e o nível mais alto, maior generalização, aquele contido no topo. A relação entre as dimensões é feita através da quantidade de vendas de um item, representada pela métrica `quantidade_vendas`. Dessa forma, o cubo, no estado apresentado, fornece informações acerca da quantidade de vendas de determinados tipos de produtos em determinados anos e cidades.

Juntamente com a representação em forma de cubo, os servidores *OLAP* fornecem alguns operadores – denominados operadores *OLAP* –, responsáveis por permitir a navegação do usuário através dos dados, fornecendo diversas visões e perspectivas do cenário

Figura 6: Cubo multidimensional



Fonte: Produzido pela autora

sendo analisado. Essa navegação é feita a partir das hierarquias previamente definidas, sobre as quais os operadores atuam.

A seguir são apresentados os operadores básicos de um sistema *OLAP*. Para exemplificação dos mesmos será considerado, como estado inicial do cubo, aquele apresentado anteriormente na Figura 6.

- **Roll-up:** realiza operações de agregação sobre os dados, usado para subir um nível na hierarquia, diminuindo o nível de detalhes da dimensão e aumentando a granularidade da mesmo, ou para diminuição de dimensão do cubo multidimensional. Na Figura 7, uma operação de *roll-up* é aplicada sobre a dimensão `DLocalidade`, agregando os dados por estado, em vez de cidade.

  Quando usado para diminuição de dimensão, uma ou mais dimensões são removidas do cubo multidimensional.

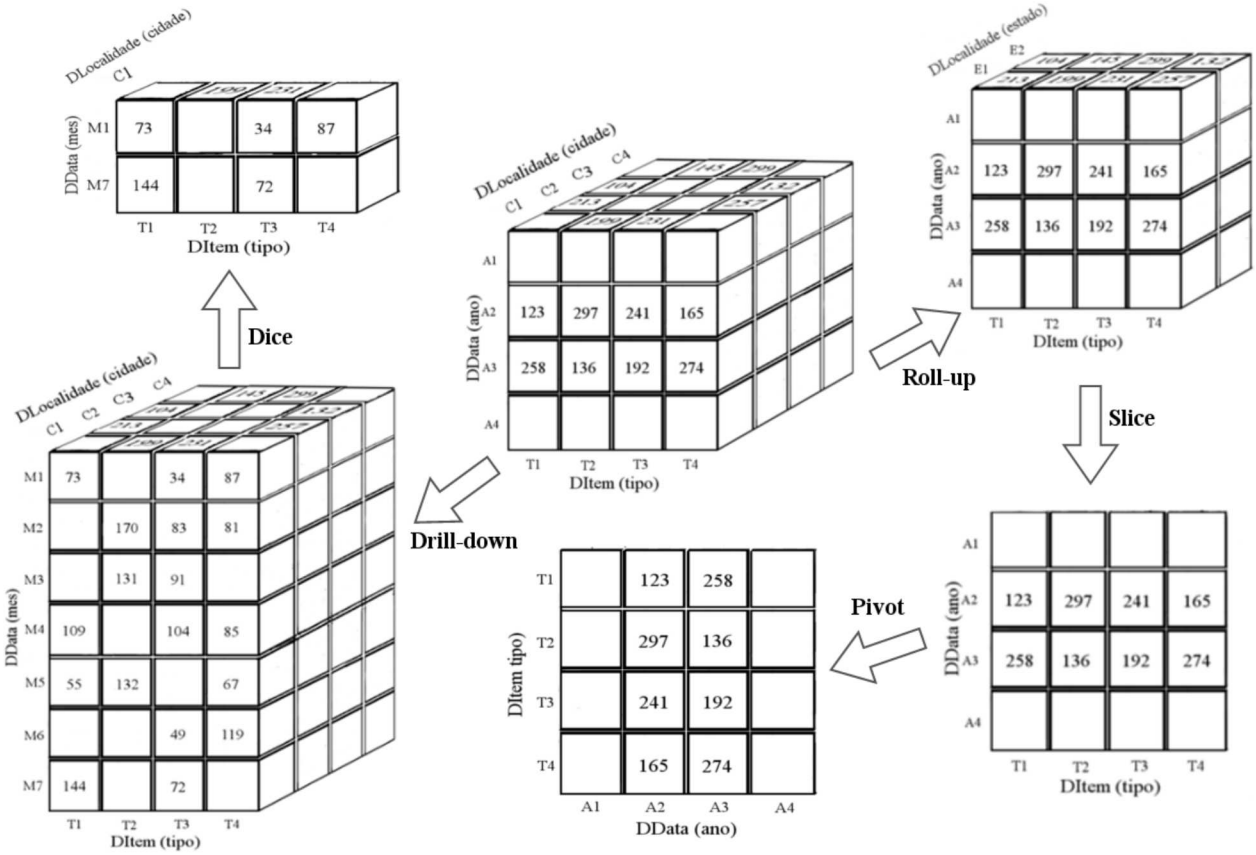
- **Drill-down:** é a operação inversa do *roll-up*, navegando de um nível mais alto da hierarquia, menos detalhado, para um mais baixo, com mais detalhes. Pode também ser utilizado para adicionar dimensões no cubo multidimensional, ao contrário do *roll-up* que é utilizado para diminuição de dimensões. Na Figura 7 é realizado um *drill-down* na dimensão `DData`, detalhando as vendas por mês no lugar de ano (por motivos de legibilidade somente parte dos dados é mostrada).

- **Slice e Dice:** a operação de *slice* realiza uma seleção sobre uma única dimensão do cubo, resultando em um subcubo do cubo original, enquanto que a operação de

dice realiza uma seleção em duas ou mais dimensões, resultando em um ou mais subcubos. Na Figura 7 é realizado um *slice* sobre o cubo resultante da operação de *roll-up*, resultando em um cubo 2D, enquanto que uma operação de *dice* é realizada sobre o cubo resultado do *drill-down*.

- **Pivot:** pivô, ou rotação, é uma operação que realiza a rotação dos eixos do cubo, mudando a posição das dimensões, com o intuito de prover uma maneira alternativa de visualização dos dados. Na Figura 7 é realizada uma rotação sobre o cubo 2D obtido a partir da operação de *slice* feita previamente.

Figura 7: Operadores *OLAP*



Fonte: Produzido pela autora

Existem ainda alguns operadores mais avançados, como o *drill-across*, que permite a execução de operações envolvendo mais de uma tabela fato, e o *drill-through*, que usa mecanismos SQL para navegar do nível mais baixo de um cubo para as suas tabelas relacionais (HAN; KAMBER; PEI, 2012).

2.2 A nova geração de *BI*

A nova geração de *BI*, conhecida também como *BI* 2.0, surgiu a partir da evolução da Web, surgimento da riqueza de informações que a mesma proporciona atualmente para a tomada de decisão e da necessidade de mecanismos de *BI* capazes de lidarem com esses dados, que são, em sua maioria, semi ou não estruturados (CHEN; CHIANG; STOREY, 2012).

Com o crescimento de forma exponencial e contínua do volume de dados da *World Web Wide* (GUPTA, 2014), e o advento da Web 2.0, a Internet tornou-se um ambiente extremamente dinâmico, onde o próprio usuário é criador de conteúdo. Dessa forma, pode-se encontrar nela informações vindas dos mais diversos lugares e pessoas, principalmente com o surgimento, e alto nível de utilização, das mídias sociais, que compartilham informações entre diversos usuários simultaneamente.

Alguns dos conceitos básicos propostos pela *BI* 2.0 são apresentados em (TRUJILLO; MATÉ, 2012) e definidos como:

- **Obtenção de dados em tempo real:** refere-se à atualização do *Data Warehouse* com novas informações o mais próximo possível do momento em que tais informações foram geradas em suas fontes, dado o cenário abordado e fatores técnicos.

  Com a dinamicidade com que novas informações surgem na atual era do conhecimento em que vivemos, a atualização periódica de dados realizada pelos mecanismos de *BI* tradicionais ocasiona em uma perda de dados significativa para a tomada de decisão. Dessa forma, a obtenção de dados em tempo real tornou-se um dos desafios para essa nova geração de *BI* (VAISMAN; ZIMÁNYI, 2012).

- **Software como Serviço (SaaS):** capacidade de realização de operações de *BI*, como consultas e obtenção de relatórios, a partir de um sistema armazenado em um servidor remoto e abstraindo a implementação do mesmo, além da capacidade de disponibilizar seus serviços de *BI* na internet.

- **Computação na nuvem:** está diretamente relacionado com o conceito de implementação utilizando *SaaS*, e se refere ao fato de implementar soluções de *BI* em serviços em nuvem, que são altamente escaláveis.

- **Colaboratividade:** se trata da capacidade de obter conhecimento através de informações provenientes de grupos descentralizados de pessoas, o que acontece frequentemente nas redes sociais, onde grupos de pessoas estão conectados em rede e compartilham informações umas com as outras, de maneira direta ou indiretamente, acerca dos mais diversos assuntos.

- **Mídias sociais:** refere-se ao uso das mídias sociais como fonte de dados, uma vez que a diversidade de informações sendo compartilhadas por intermédio das mesmas, por inúmeros tipos de pessoas, tende a enriquecer a tomada de decisão.

- **Mineração de opinião:** refere-se ao processo de descrever sentimentos ou opiniões provenientes de grupos de pessoas acerca de um determinado assunto. Dessa forma, é possível analisar as opiniões sobre o objeto e obter uma conclusão a partir delas. A análise de sentimento feita em cima de comentários de usuários na Web acerca de um determinado produto é um exemplo de mineração de opinião.

Como apontado por Abelló et al. (2015), a evolução dos mecanismos de *BI*, à qual essa nova geração se propõe, diz respeito, principalmente, à obtenção de dados em tempo real e à utilização da riqueza e diversidade de informações provenientes da Web e suas mídias sociais. Porém, como apresentado por González e Berbel (2014), servidores *OLAP* tradicionais não são projetados para lidarem com dados não estruturados, característica dos dados provenientes da Web, fazendo com que adaptações e/ou criação de operadores *OLAP* também seja um ponto importante da *BI* 2.0.

Existe, também, a questão de que dados provenientes de fontes Web, como mídias sociais, são muito voláteis para serem armazenados no *DW* (ETCHEVERRY; VAISMAN, 2012), uma vez que uma de suas principais características é a não volatilidade, tornando uma boa opção o uso de dados transitórios. Dados transitórios consistem de dados que são úteis para a tomada de decisão em um determinado momento e cenário, mas após tal análise podem ser descartados, pois não enriquecem o *DW* em relação às demais consultas.

Com relação à obtenção de dados em tempo real, geralmente são utilizadas técnicas que simulam esse comportamento. Isto se deve ao fato de que a obtenção de dados em tempo real propriamente dita não é factível, devido a interferência de vários fatores, como latência da rede e o processamento dos dados gerados. Dentre as abordagens existentes se encontra o uso de *Right-Time Data Warehouses*, que se baseiam no princípio de que os dados devem ser buscados na fonte e atualizados no *DW* conforme forem necessários (THOMSEN; PEDERSEN; LEHNER, 2008). Dessa forma, os dados se tornam disponíveis para a tomada de decisão de acordo com a necessidade do gestor e o usuário realiza a requisição dos mesmos sob demanda, fazendo com que o *DW* reflita a imagem atual das fontes de dados no momento da consulta, exibindo dados atualizados.

Por fim, temos o conceito de *Self-Service BI*, que é caracterizado pela autonomia do usuário em criar seus relatórios e consultas analíticas diretamente, quando e onde é necessário, sem a intervenção de um especialista de TI durante o processo. A utilização de dados transitórios, onde o usuário é capaz de buscar, extrair e entregar esses dados ao *DW* apenas interagindo com a aplicação, é um exemplo de tal conceito. A ideia central é

permitir que usuários leigos na área de *TI* sejam capazes de fazerem buscas e realizarem tomadas de decisões a partir de uma boa base de informações (ABELLÓ et al., 2013).

2.3 Estado da Arte

Muitas pesquisas têm sido realizadas nos últimos anos com o intuito de evoluir os mecanismos de suporte à decisão, a fim de que os mesmos sejam capazes de lidar com a alta dinamicidade e heterogeneidade das informações do cenário atual, tornando-os mais ricos e flexíveis. Essa seção apresenta alguns dos trabalhos e pesquisas mais recentes relacionados ao tema deste projeto. Ao fim, é realizada uma análise comparativa dos trabalhos apresentados.

2.3.1 *Trabalhos relacionados*

A apresentação dos trabalhos é feita partindo do menos recente ao mais atual.

Web Cube e o vocabulário *Open Cube*

Etcheverry e Vaisman (2012) propõem uma maneira de executar operadores *OLAP* sobre cubos criados a partir de dados Web, denominados *web cubes*, relacionados com cubos *OLAP* locais, sem a necessidade de integrar os dados provenientes da Web no *DW*. O principal objetivo do trabalho é a obtenção e execução de consultas de dados multidimensionais fazendo uso da Web semântica.

A criação do *Web cube* é iniciada a partir da definição por parte do usuário das informações que devem ser buscadas, que no estudo de caso utilizado durante o desenvolvimento do trabalho consiste de promoções de um determinado produto, baseado em seu preço, tempo e custo de entrega. Essas informações são consideradas as medidas da tabela fato do cubo a ser criado. Os produtos utilizados para busca foram dois modelos de câmeras fotográficas, `Canon T3` e `T3i DSLR`. Para a busca desses dados, os autores assumem que existe um catálogo com metadados de fontes Web que irá dizer onde os dados devem ser buscados, os mecanismos disponíveis e o formato dos resultados.

Após a obtenção dos dados, são identificadas suas medidas e dimensões, que podem ou não ser iguais àquelas do *DW* local, valores apropriados são inseridos para os dados faltantes e ambos os cubos, Web e local, são colocados no mesmo nível de granularidade, de maneira que possam ser devidamente integrados e operações *OLAP* sejam aplicadas.

Para integração dos dados obtidos das diversas fontes e representação do *Web cube* são utilizados arquivos RDFs (KLYNE; CARROLL; MCBRIDE, 2004) e um vocabulário proposto pelo autores, denominado *Open Cube*. Tal vocabulário se baseia na modelagem multidimensional e seus conceitos básicos são definidos como segue:

a) a classe `oc:Dimension` e um conjunto de níveis, modelados através da propriedade `oc:Level`, representam as dimensões;

b) a ordem parcial dos níveis de uma dimensão é definida através das propriedades `oc:ChilLevel` e `oc:parentLevel`, enquanto que os atributos são modelados usando `oc:Attribute`;

c) a tabela fato é representada utilizando a classe `oc:FactSchema` e o seu conjunto de níveis e medidas definidos usando as propriedades `oc:Level` e `oc:Measure`;

d) a função de agregação pra cada medida da tabela fato é definida utilizando a propriedade `oc:hasAggFunction`.

Para execução dos operadores *OLAP* é proposta a utilização de consultas SPARQL (PRUD'HOMMEAUX; SEABORNE, 2008). Para que a operação seja realizada, é necessário definir um novo esquema para o cubo, contendo as dimensões e atributos que estarão presentes no mesmo após a operação, e a criação de uma consulta SPARQL para popular esse novo cubo.

Um algoritmo geral para a execução do operador *roll-up* sobre *Web Cubes* foi apresentado. Tal operador foi utilizado para obtenção de resultados preliminares, retornando resultados em menos de 0.1 segundo, quando aplicado a um conjunto de 24.000 tuplas. O desempenho obtido foi considerado satisfatório por parte dos autores, uma vez que, pelo fato dos dados provenientes da Web serem extremamente concentrados em um determinado assunto e momento, os *Web Cubes* não tenderão a ser mais volumosos que isso.

Por fim, é feita uma demonstração de como *Web Cubes* podem ser exportados para o servidor *OLAP* Mondrian[1], de forma a ser integrado com os dados do SSD local e permitir a realização de operações *OLAP* sobre os dois.

SIE-OBI

SIE-OBI (*Streaming Information Extraction for Operational Business Intelligence*) é um *framework* proposto por Castellanos et al. (2012). O objetivo é diminuir o esforço gasto, por parte do desenvolvedor, na criação de fluxos de dados que exploram e integram informações provenientes da Web, onde o fluxo de novas informações é contínuo, e as relacionam com os dados internos da empresa, que possui um fluxo lento de novas informações, em tempo próximo do real.

A ideia é facilitar a definição de maneira declarativa de aplicações lógicas como um grafo de fluxo de dados, passando a tarefa de construir propriamente o fluxo de dados físico para o *framework*. Para isso, algumas técnicas de otimização, relacionadas ao desempenho, foram desenvolvidas, denominadas pelos autores de otimização QoX-Based. Dessa forma,

[1] Fonte: *Pentaho Community*. Acesso em janeiro de 2017. Disponível em <http://community.pentaho.com/projects/mondrian/>

o desenvolvedor da aplicação necessita apenas informar suas preferências de otimização (ele pode preferir priorizar acurácia ao custo de processamento, por exemplo).

Os passos que o desenvolvedor precisa seguir, a partir da interface disponibilizada, para criar uma aplicação utilizando o *framework* são:

1. Especificar duas ou mais fontes de dados;
2. Definir as operações lógicas que serão utilizadas na forma de um grafo de fluxo de dados. Tais operações devem abranger as etapas de extração e consultas;
3. Definir os objetivos e preferências para cada operação lógica do fluxo de dados.

Para desenvolvimento do trabalho apresentado, foram consideradas apenas fontes textuais como fontes de dados, e o fluxo de dados para execução do processo foi dividido em duas etapas básicas: extração de informação e processamento de consultas híbridas.

O componente de extração de informação (*IE - Information Extraction*) é treinado a partir da utilização de algoritmos de aprendizado de máquina, e seu principal objetivo é extrair entidades dos arquivos textuais provenientes das fontes de dados. As entidades encontradas são classificadas em dois tipos:

- **Entidades básicas:** refere-se a entidades nomeadas de maneira típica como, por exemplo, nomes de organizações, nomes de pessoas e datas.
- **Entidades baseadas em papéis:** são aquelas que têm um papel específico. Por exemplo, a data de expiração de um contrato, que não é obtida apenas extraindo qualquer data do documento, mas sim uma data com papel específico.

O treinamento desse componente é feito de maneira *offline*, a partir de uma base de documentos conhecidos, cujas características e classificação são rotuladas pelo usuário mediante o uso de uma *GUI*. No processo de rotulação, o usuário deve informar a categoria dos documentos, assim como as entidades que podem ser encontradas e outras informações de domínio relevantes. Como resultado do treinamento, são gerados modelos compostos de operadores para a extração das informações, incluindo o operador de limpeza de ruídos e o de NLP (*Natural Language Processing*).

Os modelos construídos são então aplicados na classificação e extração de informação de documentos desconhecidos, processo que pode ser feito tanto *offline*, para documentos de texto armazenados localmente, quando *on-line*, para documentos provenientes de Web.

Uma vez que todas as informações foram extraídas de ambas as fontes, é a vez do componente de processamento de consultas híbridas atuar. Tal componente é responsável por correlacionar os dados e realizar consultas que englobem tantos os dados obtidos de

documentos locais como aqueles provenientes da Web. O método proposto para obter a correlação dos dados é o de árvores hierárquicas de vizinhança (*hierarchical neighborhood trees*), a fim de tirar proveito dos níveis de hierarquias presentes nas dimensões de um *Data Warehouse*.

Fusion Cubes

No trabalho realizado por Abelló et al. (2013), é proposto um *framework* para *Self-Service BI* que visa a integração de dados transitórios, através da descoberta dinâmica de fontes úteis para um determinado assunto, com os dados estacionários. O objetivo principal é permitir ao tomador de decisão realizar essa busca e integração dos dados sem o auxílio de um especialista, além de prover mecanismos de conhecimento colaborativo, por intermédio do compartilhamento das informações extraídas.

O *framework* é dividido em 6 (seis) passos, durante os quais o usuário tem total controle das operações, sendo capaz de analisar os resultados obtidos em cada um deles e interferir no processo. Deste modo, ele pode decidir se deseja ir para o próximo passo ou retornar ao anterior para melhorar os resultados. Se escolher por realizar novamente um passo, o usuário pode melhorar os resultados definindo novos parâmetros que priorizem as características que ele considera prover melhor qualidade aos dados. Os passos são:

1. Especificação por parte do usuário de uma consulta *OLAP* que necessita da obtenção de dados transitórios.
2. Definição por parte do sistemas de fontes de dados potencialmente relevantes a consulta especificada.
3. Extração dos dados contidos nas fontes definidas.
4. Integração dos dados transitórios obtidos com aqueles contidos no *DW*.
5. Apresentação dos dados já integrados ao usuário.
6. Armazenamento ou compartilhamento, por parte do usuário, dos dados obtidos, caso ele deseje.

Para atingir o objetivo, os autores introduzem o que eles chamam de *fusion cube*, que é o principal componente da arquitetura escolhida. *Fusion cubes* são cubos multidimensionais diferenciados, que integram os dois tipos de dados – utilizando arquivos *RDF*s para a representação – e podem ser dinamicamente estendidos tanto em dimensões quanto em medidas.

Além disso, tais cubos estão relacionados com um conjunto de anotações responsáveis por armazenar informações acerca da qualidade, fonte de origem, confiabilidade e outros metadados, acerca de cada pedaço de informação contido no nele.

A arquitetura apresentada pelos autores é responsável por suportar todos os passos do *framework* proposto e contém outros 4 (quatro) importantes componentes:

- **Interface do usuário:** responsável tanto por permitir que o usuário especifique a busca a ser realizada quanto por exibir o cubo final obtido, além da interação com o usuário nas camadas intermediárias do processo.
- **Processador de consultas:** recebe consultas em formato declarativo e as converte em um código de consulta executável a ser processado. É ele que define se a consulta informada vai utilizar somente dados do *DW*, algum *fusion cube* já armazenado ou necessitará da criação de um novo *fusion cube*.
- **ETL:** realiza o processo de *ETL* definido previamente e a integração dos dados *Web* quando estes são utilizados.
- **Localizador de dados:** utilizado quando a consulta definida pelo usuário necessita da criação de um novo *fusion cube*. É responsável por, baseando-se no catálogo armazenado ou registros e ontologias externas, realizar as descobertas de fontes úteis, das quais os dados podem ser obtidos.

Por fim, é proposto um novo operador *OLAP*, denominado pelos autores de *drill-beyond*. Tal operador é entendido como uma forma de cruzar os limites dimensionais definidos pelo modelo de dados estacionário. Seu principal objetivo é dar ao usuário a capacidade de estender o cubo dimensional, seja em dimensões ou em medidas e instâncias. A extensão é feita utilizando uma *GUI*, onde o usuário seleciona, mediante *clicks* no modelo inicial, onde deseja que o cubo seja estendido e de que forma. A partir dessa seleção o operador realiza a extensão e faz a integração dos dados sobre o novo modelo multidimensional obtido.

Uma vez que o processo de gerenciar esse novo operador pode ser complexo, os autores sugerem a utilização de um sistema de recomendações, responsável por sugerir possíveis ações no modelo que já tenham sido executadas anteriormente pelo usuário, a fim de reutilizar consultas e diminuir consideravelmente a complexidade de formulação das mesmas.

Vale ressaltar que, toda a proposta apresentada no desenvolvimento deste trabalho é feita de maneira teórica e nenhuma implementação ou aval experimental foram apresentados.

Arquitetura Híbrida - *OLAP* e *OLTP*

Benker (2013) propõe uma arquitetura que integra características de sistemas *OLTP* e *OLAP*, utilizando banco de dados NoSQL, que é um modelo de banco de dados não

relacional que tem como prioridade o bom desempenho em relação a latência, e persistência poliglota. Um dos principais objetivos da proposta é prover dados para tomada de decisão no tempo certo (*right-time*).

O uso de sistemas de banco de dados não relacionais e persistência poliglota é feito com o intuito de demonstrar os benefícios da abordagem em relação àquela utilizando bancos de dados relacionais. A arquitetura apresentada é composta de três componentes. Ela visa diminuir a incompatibilidade dos dados com o modelo do domínio, assim como integrar sistemas *OLTP* e *OLAP*, com o objetivo de diminuir a latência entre a geração de uma informação e a reação em relação à mesma na tomada de decisão. Baseado nisso, os requisitos da arquitetura foram definidos como:

a) suportar persistência poliglota, mantendo o fluxo de trabalho e o modelo de dados como dois processos independentes um do outro;

b) permitir a integração de componentes em fluxos de trabalho de níveis mais altos, mas esconder os seus detalhes de implementação da camada de interface;

c) permitir a integração de processos *OLTP* e *OLAP*, no intuito de prover informações recentes para situações de tomada de decisão em *right-time*.

Os componentes que formam a arquitetura são definidos como: componente operacional, componente analítico e componente de mapeamento. Todos os três componentes são constituídos de duas interfaces de interação, podendo elas serem *OLTP* ou *OLAP*. Com o intuito de integrar cenários que utilizam tanto sistemas *OLTP* quanto *OLAP*, todos os componentes devem possuir ao menos uma interface *OLAP*.

O componente operacional é responsável, principalmente, pela realização de operações de persistência de dados, que seriam de responsabilidade dos sistemas *OLTP*, além de preparar os dados para serem utilizados pelo componente analítico. Para realizar a persistência dos dados, tal componente oferece uma interface *OLTP* que suporta a realização das quatro operações básicas (criar, ler, atualizar e deletar) sobre suas entidades, além de algumas operações relacionadas a lógica de negócio.

Além disso, esse componente também é responsável por analisar os dados transacionais relevantes e enriquecê-los com medidas e informações multidimensionais, com o intuito de entregar esses dados para o componente analítico. Esse procedimento é executado utilizando a interface *OLAP* do componente, quando solicitado pelo processo de integração de aplicativos, que é responsável por obter esses dados do componente operacional, fazer a integração e entregá-los ao componente analítico.

O componente analítico, por sua vez, é composto por duas interfaces *OLAP*, uma de entrada e uma de saída de dados. A sua função é a de realizar as tarefas que são de responsabilidade de sistemas *OLAP*, como a sumarização dos dados.

Ele recebe os dados na forma de um único conjunto de dados, como, por exemplo, um documento JSON e realiza a análise dos dados em um *Data Warehouse* utilizando um sistema de banco de dados não relacional. As operações de sumarização dos dados são realizadas utilizado o *framework* de *Map/Reduce*, que consiste de um modelo para processamento paralelo de grandes volumes de dados (DEAN; GHEMAWAT, 2008), tirando proveito da escalabilidade dos sistemas *NoSQL* para diminuir a latência.

Por fim, o componente de monitoramento, que, assim como o componente operacional, também é dividido entre funcionalidades *OLTP* e *OLAP*. O seu papel é de rastrear dados em tempo de execução dos processos transacionais e enriquecê-los com dados analíticos, de forma a entregar a informação correta, em *right-time*, quando a mesma é requerida para análise. Os eventos aos quais ele reage são eventos operacionais, como, o início de uma nova transação ou a execução de um componente, ou eventos periódicos, onde ele deve analisar um determinado processo em um determinado intervalo de tempo.

Descobrindo dimensões *OLAP* em dados semiestruturados

O estudo realizado por Mansmann et al. (2014) apresenta uma maneira de identificar, analisar e extrair propriedades de dados semiestruturados, provenientes de redes sociais, que possam ser transformadas em fatos ou dimensões dinâmicas no *DW*.

A fonte de dados não estruturados escolhida para desenvolvimento do trabalho foi o Twitter, a partir da qual os autores demonstram como extrair dimensões de seus *tweets* por meio da aplicação de métodos de enriquecimento sobre os mesmos.

Inicialmente, é feito uso de engenharia reversa pra mapear a estrutura dos *tweets*, caracterizada por ser heterogênea e semiestruturada, para a estrutura multidimensional do cubo de dados. Esse processo é feito manualmente por um especialista. Para isso, o objeto JSON que representa o *tweet* é primeiramente transformado em XML e armazenado temporariamente em um banco de dados BaseX (HOLUPIREK; GRÜN; SCHOOL, 2009), até que a etapa de transformação seja realizada. Na etapa de transformação, o especialista identifica as medidas e dimensões dos dados, com base na análise semântica dos mesmos. Após essa etapa o cubo inicial está definido, contendo tabelas fato e dimensões baseadas na estrutura dos dados provenientes do Twitter.

A descoberta de novas dimensões ou medidas é feita a partir desse cubo inicial. É feita uma análise das medidas, dimensões e hierarquias existentes a fim de descobrir outras propriedades que possam ser calculadas a partir dessas. Quando a propriedade descoberta é uma medida, nenhum ajuste na estrutura geral do cubo é necessário, uma vez que se trata de um campo atômico simples, inserido na tabela fato. Porém, quando a nova propriedade é uma dimensão ou nível de hierarquia, mudanças se fazem necessárias tanto no modelo quanto no povoamento do mesmo e manutenção dos dados.

Quando uma nova propriedade é descoberta, é definido no modelo como a mesma pode ser calculada a partir das propriedades iniciais. No trabalho apresentado, os autores usam como exemplo a descoberta de uma nova hierarquia, definida como `ranking` -> `rating` -> `popularity`, que pode ser obtida a partir das medidas `friends` e `followers` contidas na tabela fato do modelo inicial, e que representam, respectivamente, o número de amigos e de seguidores de um usuário.

A definição dos níveis dessa nova hierarquia foi feita como segue: i) `ranking` = 0.8*`followers` + 0.2*`friends`; ii) `rating` é baseado em porcentagem, onde os usuários são divididos em 100 grupos, de acordo com a sua posição e a porcentagem é atribuída ao grupo; iii) `popularity` contém apenas cinco instâncias e representa uma categorização dos usuários baseada na porcentagem atribuída ao *rating*. Quanto maior a classificação no *rating* de um usuário mais alta será a categoria de popularidade em que o mesmo se encontra.

Essa definição é então introduzida no modelo utilizando a notação x-DFM, que provê elementos gráficos para definição da propriedade, e a nova hierarquia pode então ser computada dinamicamente, de acordo com os valores dos campos primários, podendo operações *OLAP* serem aplicadas sobre a mesma.

Processo de *ETL* para Dados Abertos

Berro, Megdiche e Teste (2015) apresentam uma abordagem de processo *ETL* para *Self-Service BI*, que visa permitir ao usuário a realização de operações da maneira mais automática possível. A ideia base é transformar informações provenientes de dados abertos (*open data*) em grafos que possam ser explorados durante o processo de tomada de decisão e extensíveis para a Web semântica.

O processo apresentado é composto de três fases e recebe dados abertos semiestruturados como entrada, produzindo esquemas multidimensionais e um *Data Warehouse* como saída. As fases do processo são definidas como:

- **Fase 1:** responsável por realizar o processo de extração dos dados e transformá-los de semiestruturados para uma estrutura flexível. Tal processo realiza uma rotulação automática nos dados obtidos, onde o usuário é capaz de validar essa rotulação. Cada fonte *open data* resulta em um grafo rotulado.

- **Fase 2:** trata-se de uma integração holística em dois passos – um de correspondência semântica e similaridade e outro de integração estrutural – que pega como entrada os grafos gerados na fase 1 e os integra em um único grafo, juntamente com suas correspondências.

- **Fase 3:** propõe ao usuário, a partir do grafo gerado na Fase 2, um esquema composto de componentes multidimensionais, como dimensões e hierarquias, dando ao mesmo a capacidade de validar o modelo e incrementá-lo com componentes adicionais, como fatos e medidas de análise. É a partir desse esquema que o *DW* será alimentado.

Os autores apresentaram apenas o desenvolvimento da primeira fase do processo proposto, através da proposta de alguns algoritmos genéricos capazes de realizar a extração e transformação dos dados abertos.

A estratégia adotada para a extração dos dados foi a de rotulação e enriquecimento das informações obtidas, a fim de guiar a etapa de identificação do esquema utilizando somente o conteúdo das fontes. A rotulação é dividida em três tipos, sendo eles: rotulação inerente, rotulação topológica e rotulação semântica.

A rotulação inerente organiza os dados obtidos em uma matriz `M` de tamanho `n` x `m`, representando os tipos das células de níveis mais baixos dos dados, sendo estes numérico, rótulo, fórmula, temporal e espacial. A rotulação topológica, por sua vez, é responsável por identificar os blocos de dados contidos na matriz criada inicialmente e seus possíveis papéis no esquema multidimensional (dimensões, medidas etc). Por fim, é utilizada a rotulação semântica para informações espaço-temporais. Tal rotulação é feita a partir da utilização da criação de grafos genéricos, e tem o intuito de encontrar caminhos para ligar as entidades espaço-temporais.

Uma vez que os dados foram extraídos começa o processo de transformação dos mesmos em esquemas na forma de grafos. Essa transformação faz a conversão das rotulações feitas na etapa anterior em árvores hierárquicas. Para realizar essa conversão, duas alternativas são propostas: classificação dos conceitos hierárquicos utilizando rotulação e através de técnicas de mineração de dados. Ambas as técnicas são aplicadas respeitando três restrições pré-definidas, que são elas:

a) não podem haver caminhos cíclicos dentro de uma hierarquia, isto é, entre o nó folha e o nó raiz da hierarquia deve existir apenas um caminho possível;

b) quando existirem nós faltantes numa árvore de tamanho `k` pré-definido, os mesmos podem ser substituídos pela duplicação do valor do nó pai ou com a inserção de um novo nó de valor "Outro";

c) a altura da árvore deve ser a mesma quando percorrendo-a de todos os nós folhas até a raiz, isto é, as hierarquias devem ser balanceadas.

A partir da conversão dos dados em árvores hierárquicas é obtido o grafo final, que é definido como $G = (V, E)$, onde: V é o conjunto de vértices que representa uma estrutura (rótulos semânticos ou conceitos de árvores hierárquicas) ou dados numéricos; e E é o conjunto de arestas, representando relações entre dois vértices de estrutura ou um

vértice de estrutura e um vértice de dado numérico. Tal grafo pode então ser utilizado pelo processo de tomada de decisão para exploração dos dados abertos obtidos.

A experimentação realizada pelos autores demonstrou que as células de dados contendo dados dos tipos numérico, rótulo e temporal tiveram um alto índice de precisão por parte da abordagem na detecção das mesmas. Enquanto que, células cujos valores eram fórmulas ou dados espaciais tiveram um índice de apenas aproximadamente 50% de precisão.

Arquitetura de *Self-Service BI* e *Big Data*

Em (PASSLICK; LEBEK; BREITNER, 2017) é proposto um modelo de arquitetura que abrange características de *Self-Service BI* e *Big Data*. A proposta é feita de maneira genérica e tem como objetivo servir de referência para o desenvolvimento de modelos envolvendo ambos os conceitos, além de ser utilizado como guia para empresas que desejam avaliar sua arquitetura existente e melhorar a autonomia dos usuários. As camadas que constituem tal modelo são definidas como:

- **Preparação:** responsável pela extração e carga dos dados, que podem ser realizadas de três maneiras diversas: i) acesso direto para análises comuns; ii) acesso direto para análises em tempo real; iii) e o método que utiliza um processo de *ETL* clássico, que foi estendido de forma a permitir que a etapa de transformação seja opcional. Isso porque algumas análises de *Big Data* podem requerer a utilização de dados brutos que não foram tratados.

- **Infraestrutura de armazenamento e análise:** responsável pelo armazenamento dos dados e é constituída de dois componentes principais e um opcional. Os componentes principais consistem de um elemento de integração e um componente de refinamento de *big data*. O elemento de integração é responsável pelo armazenamento dos *data marts* e *Data Warehouse*, enquanto que o de refinamento fica encarregado de fornecer a arquitetura necessária para análise de *big data*, proveniente das fontes externas. O componente opcional, por sua vez, seria constituído de ferramentas que podem ser necessárias para a execução de operações em tempo real.

- **Semântica:** responsável por simplificar o acesso aos dados através dos múltiplos sistemas e fontes. A ideia é que ela realize acessos aos diferentes sistemas de armazenamentos de maneira unificada, provendo fácil acesso aos dados para os usuários com baixo nível de conhecimento técnico. No artigo, os autores citam a utilização de uma arquitetura orientada a serviços como sendo uma possível boa opção para a implementação dessa camada.

- **Apresentação:** responsável pela apresentação dos dados aos grupos de usuários finais. Essa camada está separada em três portais de usuários, cuja separação é realizada de acordo com a habilidade em relação à *BI* e às necessidades dos mesmos.

Existem ainda, contidos no modelo proposto, dois componentes que são considerados pelos autores como principais no intuito de suportar a característica de *Self-Service BI* no mesmo, sendo eles, componente de colaboração e um banco de dados de conhecimento.

O componente de colaboração se encontra na camada de apresentação dos dados nos dois portais que são direcionados a usuários que têm um maior conhecimento técnico de *BI*. O intuito desse componente é permitir a interação entre os usuários desses dois portais, de maneira que eles possam enriquecer – por meio de *feedbacks* e dados de suas próprias análises – as análises a serem realizadas por outros usuários.

O banco de dados de conhecimento, por sua vez, é um componente que estaria armazenado separadamente das camadas do modelo apresentadas acima. Ele seria responsável por armazenar todas as consultas que fossem executadas no sistema, assim como os seus resultados. O intuito é que esse banco armazene as consultas de forma que elas possam ser reutilizadas posteriormente, tornando a construção de consultas complexas mais fácil, quando as mesmas já tiverem sido executadas anteriormente, além da possibilidade de mostrar análises que estão relacionadas e fazer recomendações a partir de similaridade. A ideia é que esse seja um banco de auto-aprendizagem, através da utilização de algoritmos e mecanismos de aprendizado. O possível problema aqui, segundo os autores, é que para os mecanismos de aprendizado obterem resultados satisfatórios um grande volume de dados se faz necessário, o que pode ser difícil de se obter inicialmente.

O desenvolvimento do modelo apresentado foi feito de maneira totalmente teórica, a partir do estudo da literatura realizado pelos autores, tanto dos trabalhos propostos na área como dos conceitos abordados, assim como de entrevistas com uma quantidade de especialistas na área de *BI*. O propósito foi obter um *feedback* do modelo inicialmente proposto e realizar melhorias no mesmo, de forma a atender as necessidades da área e validar a aplicabilidade do mesmo.

2.3.2 *Análise Comparativa*

A fim de se obter uma visão de quais dos conceitos abordados pela nova geração de *BI* têm tido maior foco na literatura científica recente, e quais aqueles que ainda carecem de estudo mais aprofundado, foi feita uma sumarização de tais conceitos em relação aos trabalhos aqui apresentados.

Para fins de comparação, foram considerados na análise somente os conceitos da *BI* 2.0 que têm mais relevância ao trabalho aqui proposto, numa escalação de nível de relação com o mesmo. Conceitos que não fazem parte do escopo deste projeto, como computação

Tabela 1: Conceitos mais relevantes da BI 2.0

Identificador	**Conceito**
A	Obtenção de dados em *Real/Right-time*
B	Colaboratividade
C	Utilização de dados Web
D	Utilização de dados provenientes de mídias sociais
E	Análise semântica
F	Análise de opinião
G	Incorporação de dados transitórios
H	*Self-Service BI*

na nuvem e software como serviço, que foram apresentados na Seção 2.2, foram deixados de fora dessa análise.

Ao fim foram selecionados um total de oito conceitos e, para fins de legibilidade, a cada um destes foi atribuído um identificador alfabético, que pode ser consultado na Tabela 1.

Além dos conceitos propostos pela *BI* 2.0, outros dois aspectos importantes foram considerados na análise comparativa dos trabalhos apresentados: i) a proposta de criação ou adaptação de operadores *OLAP* capazes de lidarem com esse novo cenário; ii) a evidência de implementação da proposta apresentada pelos autores; representados na Tabela 2 pelas siglas EI e OP, respectivamente.

Tabela 2: Análise comparativa

Estudo	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**OP**	**EI**
Web Cube e o vocabulário *Open Cube*	✗	✗			✗		✗			✓
SIE-OBI	✗	✗			✗					✓
Fusion Cubes	✗	✗	✗		✗		✗		✓	
Arquitetura Híbrida - *OLAP* e *OLTP*	✗				✗					
Descobrindo dimensões *OLAP* em dados semiestruturados			✗	✗	✗	✗				✓
Processo de *ETL* para Dados Abertos			✗		✗			✗		✓
Arquitetura de *Self-Service BI* e *Big Data*	✗	✗	✗	✗	✗			✗		
Total	5	4	4	2	7	1	2	2	1+	4+
%	71	57	57	29	100	14	29	29	14	57

A partir da análise da Tabela 2, podemos observar que todos os trabalhos apresentados abordam o conceito de análise semântica, o que é coerente se pensarmos que a base para o surgimento dessa nova geração de *BI* é justamente a Web semântica.

Nota-se também que, a questão de obtenção de dados em tempo real (ou quase real) também tem obtido grande foco, sendo abordada em 71% dos trabalhos. Logo em seguida estão os conceitos de colaboratividade entre usuários e a utilização de dados provenientes da Web, sendo abordados em 57% dos estudos apresentados.

A utilização de dados provenientes de mídias sociais, incorporação de dados transitórios e proveniência de autonomia ao usuário – representado aqui pelo conceito de *Self-Service BI* –, por outro lado, ainda são pontos pouco explorados nos estudos abordados, aparecendo em apenas 29% deles. Seguidos da análise de opinião de usuários, que aparece em apenas um dos trabalhos apresentados (14%).

Em relação aos aspectos adicionais, pode-se verificar que somente um dos trabalhos apresentou uma proposta de criação ou adaptação de operadores *OLAP*, não apresentando nenhuma evidência de implementação da mesma. Em relação à implementação das demais propostas, um pouco mais da metade dos estudos, 57%, apresentaram evidência de terem-na realizado. Ainda assim, pode-se notar que, dentre os três trabalhos que mais se assemelham àquele sendo desenvolvido nesta dissertação, sendo eles, a Arquitetura de *Self-Service BI* e *Big Data*, o Fusion Cubes e o *Web Cube* e o vocabulário *Open Cube*, somente um apresentou alguma implementação da sua abordagem.

É focando nesses pontos ainda pouco explorados que o presente trabalho propõe o desenvolvimento de uma arquitetura genérica para SSDs, instanciável em diversos cenários. Tal arquitetura deve ser capaz de obter dados de fontes Web externas no momento em que os mesmos forem necessários, focando no uso de redes sociais como fonte dados e a utilização de dados transitórios. Ainda, um novo operador *OLAP* é proposto e implementado, capaz de manipular, identificar e integrar os dados semi ou não estruturados com aqueles do *DW*. O intuito é realizar essa identificação e integração por intermédio da análise do domínio de valores contidos nos campos dos dados Web, sem qualquer auxílio dos seus possíveis metadados durante o processo. Esse último componente é o ponto central desta proposta e se faz de suma importância, visto que os operadores tradicionais não foram projetados para tal finalidade e que, como observado acima, tem sido um aspecto pouco explorado.

2.4 Sumário

Neste capítulo foram apresentados os principais fundamentos acerca de *Business Intelligence* e Sistemas de Suporte à Decisão. O processo de *ETL* e o *Data Warehouse* são os principais componentes desses sistemas.

Com o crescimento da Web e advento da Web 2.0, a evolução desses mecanismos se torna extremamente necessária, a fim de serem capazes de manipular dados semi ou não estruturados. A obtenção de dados em tempo real, ou o mais próximo disso, e utilização de fontes Web e mídias sociais como fontes de dados podem enriquecer a tomada de decisão.

Essas necessidades geraram uma nova geração de *BI*, conhecida também como *BI* 2.0.

Dentre as pesquisas recentes apresentadas, a obtenção de dados em tempo real e realização de análise semântica sobre eles têm sido pontos bastante estudados. Por outro lado, a utilização de fontes Web e proposta de adaptação, ou criação, de novos operadores *OLAP* têm sido pouco explorados, assim como o conceito de *Self-Service BI*.

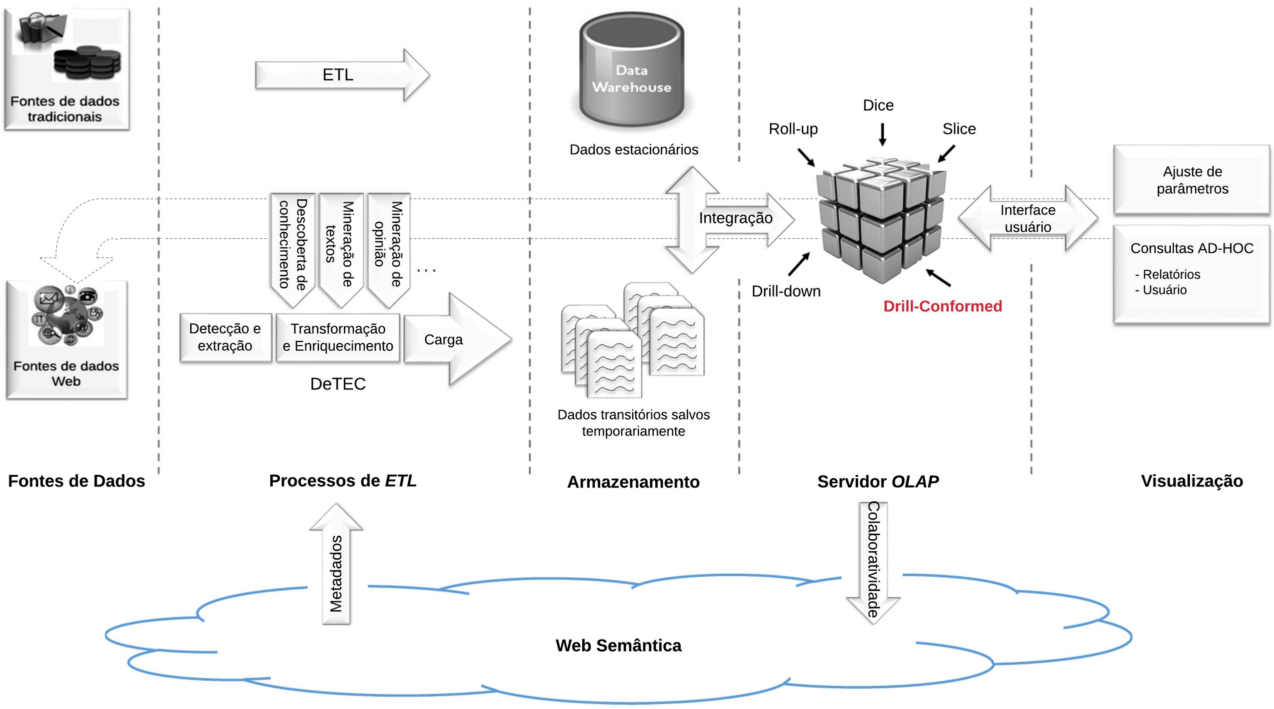
3 SelfBI e o novo operador *OLAP*

Este capítulo apresenta a arquitetura SelfBI proposta, sua modelagem e características, juntamente com o operador *OLAP* proposto, denominado *Drill-Conformed*, e suas definições.

3.1 SelfBI - A arquitetura proposta

A arquitetura proposta, denominada SelfBI e apresentada na Figura 8, é composta de cinco camadas e tem como objetivo ser uma arquitetura genérica, de integração de dados transitórios com dados estacionários, com o fim de permitir a realização de consultas utilizando dados Web e em tempo próximo ao real.

Figura 8: Arquitetura proposta



Fonte: Produzido pela autora

O eixo principal da arquitetura são as consultas centradas no usuário, a partir da demanda personalizada e ajustes de parâmetros. As consultas podem requisitar tanto dados transitórios como estacionários, ou ambos. Cada uma de suas camadas pode ser definida como segue abaixo e as principais serão melhor descritas nas seções subsequentes.

- **Fontes de dados:** estudo e seleção das fontes de dados úteis ao cenário sob o qual a arquitetura será aplicada.
- **Processos de *ETL*:** extração das fontes, limpeza, transformação e carga dos dados.
- **Armazenamento:** armazenamento dos dados obtidos pela *ETL* nos modelos de dados estacionários e transitórios previamente definidos.
- **Servidor *OLAP*:** processamento das consultas multidimensionais e disponibilização dos operadores *OLAP* para utilização do usuário.
- **Visualização:** ferramentas para interação com o usuário e exibição dos resultados ao mesmo.

Além disso, como pode ser visto na imagem, duas das camadas interagem com a Web semântica durante sua execução. A primeira é a camada de *ETL*, que, durante o processo de DeTEC - (D)etecção e (e)xtração, (T)ransformação, (E)nriquecimento e (C)arga, busca obter informações semânticas acerca dos dados transitórios obtidos. Esse processo é realizado a fim de ter conhecimento das possíveis transformações e enriquecimentos a serem feitos, e como serem feitos. Além de detectar os atributos que são possíveis métricas em relação ao que o usuário deseja analisar. A segunda é a camada do servidor *OLAP*, onde o operador proposto, *Drill-Conformed*, disponibiliza as informações que ele conseguiu extrair do domínio dos dados transitórios na Web semântica, a fim de disponibilizar metadados que possam colaborar com os demais usuários que venham a lidar com esses domínios de dados.

3.1.1 Fontes de dados

Essa é a etapa inicial de instanciação da arquitetura e consiste na seleção de fontes de dados que provêm informações úteis para o contexto ao qual a mesma será aplicada. Tal seleção é feita previamente de forma manual pelo gestor do SSD e se baseia em critérios como: utilidade e relação com o cenário proposto, disponibilidade e facilidade de obtenção dos dados.

Para a arquitetura proposta, faz-se necessária a seleção de dois tipos de fontes de dados, sendo eles:

- **Fontes estacionárias:** de onde são extraídos dados que serão permanentemente armazenados no *DW*, como, por exemplo, a base operacional da instituição à qual a arquitetura será aplicada.
- **Fontes transitórias:** fontes Web que fornecem os dados úteis à tomada de decisão em um determinado momento e em relação a um assunto específico.

3.1.2 Armazenamento

Apesar da etapa de armazenamento estar localizada após aquela de *ETL* na arquitetura proposta (e nos Sistemas de Suporte à Decisão), a sua definição é sempre feita antecipadamente. Isso se deve ao fato de que é através da especificação do modelo de dados do *DW* que se pode definir o funcionamento das etapas do processo de *ETL*, como, por exemplo, quais dados devem ser extraídos, como eles devem ser padronizados e de que forma (e onde) será realizada a carga dos mesmos.

Nessa etapa, é realizada a definição e implementação dos modelos de dados utilizados para armazenamento dos dados. Devido às características da arquitetura proposta, dois tipos de armazenamento de dados necessitam ser criados, um para armazenamento persistente dos dados estacionários e outro para armazenamento temporário dos dados transitórios.

O modelo de dados estacionários segue a estrutura de um *DW* tradicional. O de dados transitórios, em contrapartida, visa ser um armazenamento do tipo temporário, onde os dados ficarão armazenados para realização das consultas por parte do usuário até que seu tempo de vida útil seja atingido e eles sejam descartados. O tempo de vida útil dos dados transitórios é definido de acordo com o contexto onde a arquitetura é aplicada. Pode ser somente enquanto o usuário que o requisitou realiza suas consultas, ou pode ser definido um período pré-determinado pelo gestor do sistema, a partir de análises realizadas pelo mesmo. A única restrição é que não deve ser definido um período considerado longo ao cenário, o que anularia o propósito da utilização de dados transitórios.

Por esses dados se tratarem de dados Web, em sua maioria semi ou não estruturados, são boas opções de escolha para armazenamento mecanismos que permitam flexibilidade no esquema dos mesmos, como sistemas de banco de dados NoSQL. Os sistemas de banco de dados NoSQL (largamente referido como "*Not Only SQL*") são sistemas que não utilizam a modelagem relacional. Suas principais características são serem livre de esquema, altamente distribuíveis e operáveis em hardware básico. Dessa forma, informações podem ser armazenadas em sua estrutura base – sem a necessidade de modificações complexas para corresponder ao modelo de dados – e a inserção e recuperação é feita de maneira simples, uma vez que não existem restrições de relacionamento entre os dados. Consequentemente, tais mecanismos se tornam uma boa alternativa para sistemas que lidam com grandes volumes de dados e estruturas diversas. Vale ressaltar que, a despeito dos benefícios apresentados, sistemas NoSQL geralmente abrem mão de uma ou mais propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), presentes em sistemas de gerenciamento de banco de dados relacionais, permitindo a inserção de valores duplicados e inconsistências (FOWLER, 2015).

3.1.3 Processos de *ETL*

Esse é um processo de extrema importância em Sistemas de Suporte à Decisão, responsável pela extração dos dados de suas fontes, limpeza e transformação dos mesmos – a fim de padronizá-los – e cargas destes no *Data Warehouse*.

Na arquitetura proposta, faz-se necessária a criação de dois processos de *ETL* distintos, como pode ser visto na Figura 8. O primeiro é executado sobre os dados estacionários e o segundo para o tratamento dos dados transitórios. A *ETL* dos dados estacionários segue um processo de *ETL* tradicional, enquanto que a de dados transitórios foi denominada de DeTEC - Detecção e extração, Transformação, Enriquecimento e Carga dos dados. Esse último é devidamente explicado na seção subsequente.

3.1.3.1 DeTEC - Detecção e extração, Transformação, Enriquecimento e Carga

O processo de *ETL* dos dados transitórios possui diferenças em relação a um processo tradicional e, por isso, foi denominado DeTEC no presente trabalho. A etapa de extração é também responsável pela detecção de dados úteis e uma etapa de enriquecimento dos dados foi adicionada. Além disso, em comparação ao processo utilizado pelo *DW* estacionário, que é executado periodicamente a fim de atualizar os dados do *DW* com os novos dados inseridos em suas fontes, esse é um processo contínuo, executado a cada vez que o usuário realiza uma nova consulta que requeira a obtenção de dados transitórios. Cada uma das etapas do processo é definida a seguir.

Detecção e extração

Essa é a etapa responsável por buscar, nas fontes previamente escolhidas, dados relacionados ao assunto sendo pesquisado pelo usuário. A detecção consiste em criar um mecanismo que possa ser utilizado para identificar, nas fontes de dados Web, a existência de dados úteis à consulta sendo realizada. Posteriormente, o passo de extração é encarregado de obter todos os dados que foram encontrados na etapa de detecção e trazê-los para o SSD. Uma vez que todos os dados sejam extraídos, o processo de transformação é acionado.

Transformação

O objetivo dessa etapa é similar àquele de um processo de *ETL* tradicional: ser responsável pela transformação e padronização dos dados obtidos para que eles possam ser devidamente inseridos no mecanismo de armazenamento. As transformações a serem aplicadas depende dos dados sendo utilizados. A identificação destas pode ser feita a partir da interação com a Web semântica, obtendo metadados e informações colaborativas acerca dos dados, disponibilizadas por processos que já fizeram algum tipo de processamento sobre essas mesmas fontes.

Enriquecimento

O enriquecimento a ser realizado por essa etapa trata-se de um enriquecimento semântico. Esse tipo de enriquecimento é caracterizado pela a agregação de conhecimento semântico a campos que inicialmente não continham informação semântica sobre si. O tipo de análise semântica específica a ser realizada é definido de acordo com o contexto ao qual a arquitetura é aplicada. Um exemplo é a análise de sentimento, que pode ser realizada sobre campos do tipo texto, a fim de identificar a opinião de um usuário e/ou *feedback* acerca de algum assunto. Extração de entidades, palavras-chave e detecção de eventos também são exemplos de análises semânticas que podem ser executadas sobre dados (MANSMANN et al., 2014).

Em termos da arquitetura proposta neste trabalho, é também nessa fase que os atributos transitórios que possuem características de métricas de análise para a tomada de decisão devem ser identificados. Esse processo é auxiliado pela Web semântica.

Carga

O processo de carga tem a mesma tarefa daquele de uma *ETL* tradicional: carregar os dados obtidos nos mecanismos de armazenamento previamente definidos.

3.2 Drill-Conformed - O novo operador *OLAP*

Nessa seção são descritos a definição, objetivos, restrições e desenvolvimento do operador *OLAP* proposto: *Drill-Conformed*.

3.2.1 Objetivos

O operador proposto é um operador *OLAP* que visa realizar a integração de dados contidos no *Data Warehouse* com dados provenientes da Web e disponibilizar novas informações para os usuários, da maneira mais automática possível.

O seu principal objetivo consiste em realizar essa integração dos dados sem ter nenhum conhecimento prévio de suas estruturas, por meio da análise somente do domínio de valores dos atributos obtidos, quando os metadados não estão disponíveis ou não é possível fazer uso dos mesmos. Dado isto, os objetivos específicos do operador podem ser definidos como:

1. Integrar dados Web com os dados pertencentes ao *Data Warehouse* de maneira autônoma, concisa e coerente, utilizando somente o domínio de valores (sem conhecimento prévio da semântica dos dados obtidos ou acesso aos seus metadados).

2. Prover autonomia ao usuário, de forma que o mesmo esteja apto a realizar consultas de maneira independente e utilizando ambos os dados, sendo capaz de interagir com o sistema para obter os resultados desejados sem a necessidade de intervenção de um especialista em *TI*.

3. Disponibilizar informações que possam ser úteis à tomada de decisão mas não fazem parte dos dados armazenados no *Data Warehouse*.

4. Realizar a integração dos dados em tempo próximo do real.

5. Colaborar com a Web semântica por meio do compartilhamento de informações descobertas.

3.2.2 Restrições de escopo

Duas restrições de escopo foram definidas quando da elaboração da proposta:

1. O idioma dos dados contidos no *Data Warehouse* e dos dados transitórios deve ser o mesmo.

2. O operador não lida com dados transitórios estritamente numéricos, a menos que os mesmos sejam apontados como métricas de análise pelo processo de DeTEC. A análise de domínio de dados estritamente numéricos sem algum contexto, como a utilização dos metadados, por exemplo, é de difícil realização, uma vez que números por si só não têm um significado semântico.

3.2.3 Definições

O operador é baseado na proposta apresentada em (ABELLÓ et al., 2013). O nome escolhido representa a exploração dos dados nas consultas, que tem de ser projetada de maneira coerente, de forma a compartilhar uma estrutura uniforme que permita a fusão desses dados como um todo integrado. Uma das características chaves do Drill-Conformed é o fato dele trabalhar apenas com o domínio de valores dos atributos de dimensões do modelo multidimensional.

O conjunto de todos os atributos pertencentes ao modelo estacionário é aqui representado por $E = (Ae_1, Ae_2, Ae_3, Ae_4, ..., Ae_l)$, onde $l \in \mathbb{N}^*$ é o tamanho do conjunto. Simetricamente, $T = (At_1, At_2, At_3, At_4, ..., At_k)$, sendo $k \in \mathbb{N}^*$ o tamanho do conjunto, é o conjunto de todos os atributos extraídos dos dados transitórios obtidos. Existe ainda o conjunto D, que representa todas as dimensões do modelo estacionário e cujos elementos são identificados como De_d, onde $m \in \mathbb{N}^*$ é o tamanho do conjunto e $1 \leq d \leq m$. A função que representa o domínio de valores de um determinado atributo ou dimensão x é

representada por $dom(x)$, enquanto que o valor contido em um determinado atributo y é apontado por $val(y) \in dom(y)$.

Por definição, para um atributo ser parte do *Data Warehouse*, ele deve estar inserido em alguma dimensão do mesmo. Dessa forma, todos os atributos estacionários pertencem tanto ao conjunto E como fazem parte de alguma dimensão $De_d \in D$. A notação aqui utilizada para dizer que um atributo estacionário está contido em uma determinada dimensão de D foi Ae_i^d, onde i indica qual o atributo $Ae_i \in E$ e d a dimensão $De_d \in D$ à qual ele pertence.

Por fim, existem duas relações que podem ser definidas entre um At_j e um elemento do *Data Warehouse*, sendo aqui definidas como RM e RI. $RM(At_j, x)$ indica que o atributo transitório At_j possui uma relação de mapeamento com o elemento x dos dados estacionários, onde $x \in D \cup E$. Uma relação de mapeamento entre dois elementos indica que o domínio de valores de ambos tem o mesmo significado semântico. $RI(val(At_j), val(Ae_i))$, por sua vez, indica que $val(At_j)$ e $val(Ae_i)$ são equivalentes e devem ser integrados.

As premissas[1] básicas sobre as quais o operador é desenvolvido são dadas como segue.

Premissa 1: a estrutura e semântica dos dados é desconhecida.

Premissa 2: a única informação em relação à semântica dos dados transitórios passada para o operador é quais atributos (At_j) não devem ser considerados como possíveis dimensões durante o mapeamento, pois foram anteriormente identificados, pelo processo de DeTEC, como objeto de análise da tabela fato e, portanto, serão integrados ao modelo como métricas.

Premissa 3: os tipos de dados que o operador recebe são do tipo numérico, data e texto livre. Outros tipos de dados não estruturados, como imagens e vídeos, não são tratados pelo operador nessa primeira elaboração.

Visando a integração dos dados e adição de informações úteis à tomada de decisão, foram definidas duas classes de atributos provenientes dos dados transitórios, $Ci \subset T$ e $Cr \subset T$, onde $Ci \cap Cr = \emptyset$, definidas como:

- **Atributos de intersecção (Ci):** atributos transitórios que servem para fazer o cruzamento com os dados estacionários.

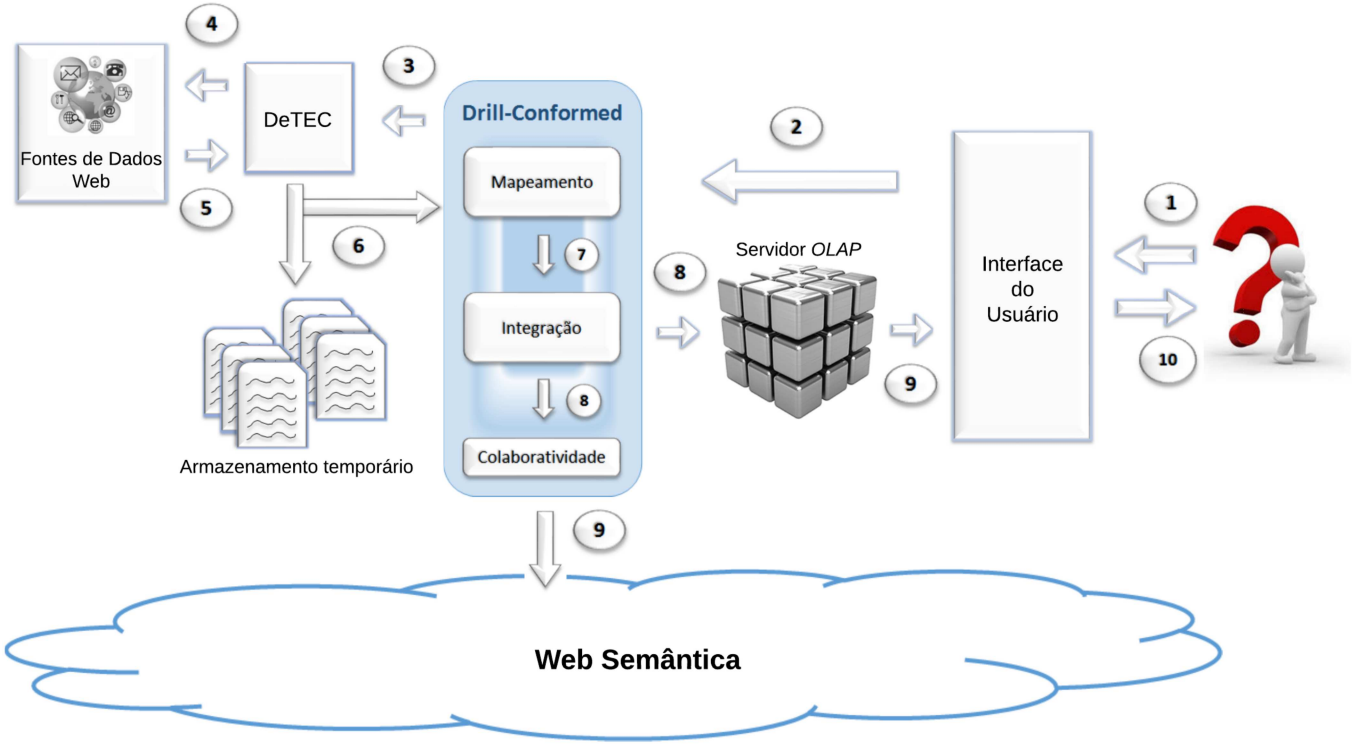
- **Atributos relevantes (Cr):** atributos transitórios que não fazem parte do modelo de dados estacionário (e por isso não pertencem à primeira classe) mas podem enriquecer a tomada de decisão.

[1] Chama-se de premissa aquilo que a autora assumiu como verdadeiro para iniciar o desenvolvimento do projeto.

Os detalhes e definição de ambos os processos, classificação dos atributos e integração dos dados, são expostos nas Seções 3.2.4 e 3.2.5, respectivamente. Na Seção 3.2.6 é explicado como as informações obtidas do processo de mapeamento e integração dos dados pode colaborar com a Web semântica.

O operador é acionado sempre que o usuário realiza uma consulta que envolve a obtenção de dados transitórios. A Figura 9 apresenta o operador e seus componentes, juntamente com o fluxo de informações resultante de uma consulta envolvendo dados transitórios.

Figura 9: Drill-Conformed - Fluxo de informações



Fonte: Produzido pela autora

Como pode ser observado, o ciclo de informações do momento em que o usuário realiza tal consulta até que os dados sejam retornados a ele é composto de 10 passos, executados da seguinte maneira:

- **Passo 1:** o usuário submete uma consulta que inclui dados Web.
- **Passo 2:** a interface notifica o operador Drill-Conformed de que uma nova consulta envolvendo a utilização de dados transitórios foi efetuada, passando para ele os parâmetros informados pelo usuário.

- **Passo 3:** o operador aciona o processo de DeTEC dizendo quais dados Web devem ser obtidos e informando os parâmetros que foram passados a ele.
- **Passo 4:** o processo de DeTEC inicia o processo de detecção e extração dos dados das fontes Web previamente selecionadas;
- **Passo 5:** os dados Web são obtidos e o processo de DeTEC executa as etapas de transformação e enriquecimento.
- **Passo 6:** o processo de DeTEC realiza a carga dos dados e notifica o operador de que seu trabalho foi finalizado, acionando o componente de mapeamento.
- **Passo 7:** o operador finaliza a etapa de mapeamento dos dados e passa o resultado obtido para o componente de integração.
- **Passo 8:** o componente de integração identifica onde os dados devem ser integrados no *Data Warehouse* e realiza a integração dos mesmos, tornando-os acessíveis ao servidor *OLAP*. Simultaneamente, as informações descobertas por intermédio do operador, acerca da estrutura e semântica dos dados, são passadas ao componente de colaboratividade, para que ele possa processá-las.
- **Passo 9:** a interface do usuário é notificada de que os dados estão disponíveis. Simultaneamente, o componente de colaboratividade do operador envia as informações para a Web semântica.
- **Passo 10:** o usuário pode visualizar o resultado da consulta, utilizando-os para a tomada de decisão.

3.2.4 Componente de Mapeamento

O componente de mapeamento é responsável por, a partir da análise de domínio de valores de cada atributo dos dados estacionários e daqueles obtidos da Web, separar os dados transitórios nas duas classes apresentadas na Seção 3.2.3: atributos de intersecção e atributos relevantes.

Para classificar cada At_j e identificar com qual atributo do *Data Warehouse* ele pode ser mapeado, foi feito uso de técnicas probabilísticas e matemáticas, com o intuito de, para cada atributo $At_j \in T$, obter o atributo $Ae_i \in E$ que maximiza a coocorrência entre $dom(Ae_i)$ e $dom(At_j)$. O processo de mapeamento dos atributos foi dividido em duas etapas: i) identificar em qual dimensão De_d cada At_j deve ser mapeado, se houver; ii) identificar, para cada mapeamento obtido na etapa anterior, para qual atributo Ae_i^d, pertencente à dimensão De_d obtida, deve ser realizada a integração. Em ambas as etapas, o intuito é identificar o elemento (dimensão ou atributo, dependendo da etapa) que maximiza a probabilidade de ocorrência do $dom(At_j)$.

O principal motivo de dividir o mapeamento em duas etapas foi o tamanho da matriz esparsa gerada, o espaço em memória requerido e o impacto destes na eficiência do algoritmo. Note que, matemática e logicamente falando, o mapeamento poderia ser realizado em uma única etapa, onde o componente tentaria identificar diretamente o atributo Ae_i para cada atributo At_j. Porém, como é já sabido, *Data Warehouses* tendem a possuir grandes volumes de dados e uma quantidade considerável de atributos. Assim sendo, a esparsidade e o espaço em memória requerido para carregar o número de ocorrências (ou mesmo as probabilidades já calculadas) de cada palavra, para cada um dos atributos Ae_i, tende a explodir conforme o DW cresce, tornando necessário hardware mais poderoso ou acesso direto a disco. A utilização de duas etapas não garante que isso não irá acontecer em algum momento durante a vida útil do DW, mas retarda o processo quando em comparação à utilização de uma única etapa. Por último, com os valores dos atributos estacionários divididos em um número maior de domínios, a chance de haver colisão entre um ou mais desses se torna maior, o que diminui a capacidade do algoritmo de identificar corretamente o mapeamento de um At_j.

Os métodos probabilísticos que o operador usa para identificar o mapeamento de um At_j são definidos da seguinte maneira: seja W o conjunto de termos individuais[2] pertencentes a $val(At_j)$ para um determinado At_j e $S = \{D \cup E\}$ o conjunto de elementos para o qual esse At_j pode ser mapeado. Descobrir o elemento $s \in S$ para o qual a probabilidade de ocorrência de $val(At_j)$ é maior, consiste em encontrar o elemento s que maximiza o produtório:

$$\prod_{w}^{W} p(w, s) \tag{3.1}$$

sendo $p(w, s)$ a função que calcula a probabilidade do termo w ocorrer no elemento s em função do número de vezes que o mesmo ocorre no conjunto S como um todo, definida como segue:

$$p(w, s) = \frac{n(w, s)}{n(w)} \tag{3.2}$$

onde, $n(w, s)$ é o número de vezes em que w ocorre no elemento s e $n(w)$ é a função que conta o número de vezes que w ocorreu no conjunto S como um todo, definida como:

$$n(w) = \sum_{s}^{S} n(w, s) \tag{3.3}$$

Analisando o produtório apresentado na Equação 3.1, é possível observar que, nos casos em que $\exists w \in W | p(w, s) = 0$ para algum $s \in S$, o produtório inteiro será zerado,

[2] Neste manuscrito, refere-se como termos individuais de $val(At_j)$ ao conjunto de termos obtidos da tokenização de $val(At_j)$ a partir de um conjunto de delimitadores pré-definidos.

anulando a probabilidade de ocorrência do conjunto W para esse s. Para evitar que isso aconteça, a suavização de Laplace (também conhecida como *Add-one smoothing*) (BOODIDHI, 2011) é aplicada na função $p(w, s)$, impedindo que o produtório seja anulado. Dessa forma:

$$p(w, s) = \frac{n(w, s) + 1}{n(w) + |S|} \tag{3.4}$$

onde, $|S|$ é o número de termos distintos presentes no conjunto S.

Existe ainda, em operações em ponto flutuante realizadas por mecanismos computacionais, o problema de instabilidade numérica, que consiste no efeito de propagação de erros, caracterizado pelo aumento significativo do erro relativo dos resultados parciais, a partir da qual o resultado final pode se tornar extremamente errado. Uma solução que pode ser aplicada a esse problema, quando se tratando do cálculo de probabilidades, consiste em representar as probabilidades no espaço logarítmico. Seguindo a propriedade matemática de que $\log(x * y) = \log(x) + \log(y)$ e considerando o fato de que *log* no intervalo $[0, 1)$ é negativo, o produtório apresentado em 3.1 é modificado para:

$$\sum_{w}^{W} -\log(p(w, s)) \tag{3.5}$$

fazendo com que o problema que inicialmente consistia em encontrar o elemento s que maximizasse o produtório apresentado na Equação 3.1 se torne um problema de minimização da Equação 3.5.

O Algoritmo `getDimensionsMap` apresenta o pseudocódigo da rotina criada para execução da primeira etapa do mapeamento. Tal rotina é responsável por, para cada $At_j \in T$, encontrar uma dimensão $De_d \in D$ que minimize a Equação 3.5 para os valores do $dom(At_j)$. Para isso, ele utiliza um sistema de votos com pesos que funciona da seguinte forma:

a) para cada atributo At_j, a dimensão De_d que minimiza a somatória da Equação 3.5 para o conjunto de termos de $val(At_j)$ ganha o voto (linhas 30 à 35);

b) o peso do voto é definido como o inverso da probabilidade obtida na minimização, de forma a garantir que, quanto menor o valor da mesma, maior influência tem o voto (linha 33);

c) por estarmos utilizando a suavização de Laplace, antes de atribuir o voto a uma dimensão, é verificado também se a probabilidade obtida não é equivalente aos valores apenas da suavização, isto é, que não há nenhuma ocorrência das palavras e o valor obtido é puramente o resultado da suavização aplicada sobre cada uma delas (linha 31);

d) ao fim, quando todos os votos foram dados, a dimensão que obteve um maior número de votos em peso é considerada a dimensão cujo mapeamento do atributo deve ser realizado;

e) para cada valor de mapeamento obtido, o mesmo só é considerado se o número de votos em quantidade da dimensão escolhida não for inferior a porcentagem de votos mínima (`min_votes`) passada como parâmetro para o algoritmo (linhas 40 à 46). Isso impede que votos esporádicos definam o resultado do mapeamento.

A entrada requerida pelo algoritmo consiste no conjunto de atributos que foram anteriormente identificados como objetos de análise da tabela fato (`metrics`) e, por isso, não devem fazer parte do mapeamento, os dados transitórios obtidos (`situational`), o corpora contendo os valores do *Data Warehouse*, divididos por dimensão (`D_corpora`), o conjunto de atributos transitórios extraídos dos dados obtidos (`T`) e a quantidade mínima de votos que uma dimensão deve ter para que seu mapeamento seja considerado válido (`min_votes`). Como saída, o algoritmo retorna um conjunto de pares $<At_j, De_d>$, indicando a dimensão De_d em que um determinado atributo transitório At_j deve ser mapeado (`mapping`).

Para realizar os cálculos de probabilidade em relação a um atributo At_j de um documento, o algoritmo primeiramente tokeniza o $val(At_j)$, representado pela função `getNoStopWords` (linha 18), obtendo o seu conjunto de termos e eliminando *stopwords*. Posteriormente, para cada dimensão De_d é aplicada a função `getMinusLogProbability` (linhas 24 à 28) em relação a cada um dos termos obtidos, de forma a obter o valor de $-\log(probabilidade)$ para cada termo em cada dimensão. Simultaneamente, as probabilidades de cada termo em relação a cada De_d vão sendo somadas (linha 25) – Equação 3.5 –, cujo resultado da soma é o que define qual dimensão minimiza a $-\log(probabilidade)$ do conjunto de valores (linha 30).

Os atributos At_j para os quais não foi encontrada nenhuma dimensão a ser mapeada nessa primeira etapa são classificados como atributos relevantes. Os demais atributos são tidos como atributos de interseção e passados para a segunda fase do mapeamento. O pseudocódigo da segunda etapa do mapeamento, aquela responsável por identificar qual atributo Ae_i, dentre os pertencentes à dimensão De_d encontrada na primeira etapa, deve ser utilizado na integração com o At_j correspondente, é apresentado pelo Algoritmo `getDimensionAttrMap`.

Da mesma forma que o algoritmo da primeira etapa, esse algoritmo também utiliza a minimização da $-log(probabilidade)$ para identificar o atributo a ser mapeado. A diferença se encontra na forma como o mapeamento é decidido. Uma vez que as regras de votação da primeira etapa têm como intuito oferecer um grau de confiança de que o campo se encontra na dimensão escolhida, não é necessário fazer esse tipo de verificação na segundo etapa. Isto porque, quando analisando, se o atributo At_j tem um cruzamento com a dimensão

Algoritmo getDimensionsMap

Entrada: metrics[], situational[], $D\_corpora$, T, min_votes
Saída: mapping[]

```
1: (D, corpora) ← getDimensionsInfo(D_corpora)
2: qty_attr ← len(T)
3: qty_dimensions ← len(D)
4: //Obtém o número de palavras distintas contidas juntando todas as dimensões
5: vocabulary_size ← getVocabularySize(corpora)
6: //calcula qual é o valor da -log probabilidade com suavização
7: //de Laplace quando a palavra não tiver ocorrências
8: laplace_only ← −log(1/vocabulary_size)
9:
10: //matrizes que armazenam os votos obtidos por cada atributo At_j
11: //em relação a cada dimensão De_d
12: votes_with_weight ← initZerosArray(qty_dimensions, qty_attr)
13: qty_votes ← initZerosArray(qty_dimensions, qty_attr)
14:
15: for data in situational do
16:   for At_j ∈ T and At_j ∉ metrics do
17:     //Retorna um array contendo todos os termos de At_j que não são stop words
18:     data_words ← getNoStopWords(data[At_j])
19:     qty_words ← len(data_words)
20:
21:     //Inicializa com 0s o array que armazena a soma das -log prob de cada De_d
22:     probs ← initZerosArray(qty_dimensions)
23:
24:     for De_d in corpora do
25:       for word in data_words do
26:         probs[De_d] += getMinusLogProbability(corpora, word, De_d)
27:       end for
28:     end for
29:
30:     //Obtém a dimensão que minimiza a probabilidade e o valor da mesma
31:     (De_d, prob_value) ← getMinProb(probs)
32:     if prob_value <> laplace_only * qty_words
33:       //quanto menor for a probabilidade maior será o peso do voto
34:       votes_with_weight[De_d][At_j] += 1/prob_value
35:       qty_votes[De_d][At_j] += 1
36:     end if
37:
38:   end for
39: end for
40:
41: min_votes ← len(situational) * perc_min_votes/100
42: for At_j in T do
43:   De_d ← getDimensionMaxWeight(votes_with_weight, At_j)
44:   if qty_votes[De_d][At_j] ≥ min_votes
45:     mapping[At_j] ← De_d
46:   end if
47: end for
48: return mapping
```

Algoritmo getDimensionAttrMap

Entrada: situational[], At_j, attrs_corpora
Saída: Ae_i^d

```
1: //Esse algoritmo será executado para cada par (At_j, De_d) encontrado na etapa 1
2: (Ae^d, hierarchy, qtd_attrs) ← getAttrsInfo(attrs_corpora)
3:
4: //junta todos os valores do campo a ser mapeado em uma única string
5: values ← ""
6: for data in situational do
7:    values ← concat(values, " ", data[At_j])
8: end for
9:
10: words ← getNoStopWords(values)
11:
12: //Inicializa com 0s o array que armazena a soma das -log prob para cada Ae_i^d
13: probs ← initZerosArray(qtd_attrs)
14:
15: for Ae_i^d in Ae^d do
16:    sum ← 0
17:    for word in words do
18:       sum += getMinusLogProbability(attrs_corpora, word, Ae_i^d)
19:    end for
20:    probs[Ae_i^d] ← sum
21: end for
22:
23: //Retorna um array contendo os atributos que minimizam a probabilidade
24: attrs ← getAttrsMinProb(probs)
25: if len(attrs) > 1
26:    Ae_i^d ← getMinNivelHierarchy(hierarchy, attrs)
27: else
28:    Ae_i^d ← attrs[0]
29: end if
30: return  Ae_i^d|RM(At_j, Ae_i^d)
```

De_d, obrigatoriamente esse cruzamento deverá ser realizado com algum dos atributos Ae_i pertencentes a esta dimensão (definidos como Ae_i^d). Dessa forma, não é necessário verificar um número mínimo de votos, por exemplo, uma vez que é sabido que o campo de cruzamento está nessa dimensão.

Outra diferença em relação ao algoritmo da primeira etapa é que, uma vez que se está verificando apenas um atributo At_j para o mapeamento, e não existem as restrições da quantidade de votos, não é preciso percorrer item por item dos dados transitórios. Dessa forma, a abordagem que o algoritmo usa é juntar todos os valores do atributo At_j em um único elemento e verificar a coocorrência do $dom(At_j)$ como um todo.

Os parâmetros que o algoritmo recebe são, um atributo At_j que pertence à classe

de intersecção, os dados transitórios, representados por `situational`, e o corpora dos atributos Ae_i pertencentes à dimensão De_d associada ao atributo At_j na primeira etapa. Tal corpora é composto de documentos que contêm o $dom(Ae_i^d)$ para todo Ae_i^d. Como resultado, o algoritmo deve retornar o atributo Ae_i^d cujo At_j passado deve ser cruzado.

A primeira etapa do algoritmo concatena todos os valores dos dados transitórios do atributo At_j em um único elemento, que é tokenizado eliminando as *stopwords*, da mesma forma que acontece na etapa 1 (linhas 6 à 10). Uma vez que os termos foram obtidos, o cálculo das probabilidades é realizado da mesma forma que no algoritmo anterior, através da utilização da função `getMinusLogProbability` e realizando a soma das probabilidades obtidas para cada termo em relação a cada atributo Ae_i^d (linhas 15 à 21).

Ao fim, o atributo Ae_i^d cuja soma das probabilidades minimiza a Equação 3.5 é retornado pelo algoritmo. Caso haja mais de um Ae_i^d que minimize a probabilidade, aquele de menor nível na hierarquia da dimensão De_d é utilizado (linhas 24 à 29). A escolha de retornar o menor nível na hierarquia quando há empate faz sentido devido a estrutura de um *DW*. Isto porque, se você tem informação ligada ao nível `n` da hierarquia, é possível navegar para todos os níveis acima de `n` através dos operadores *OLAP* utilizando funções de agregação, já o contrário nem sempre é válido, dependendo da abordagem utilizada na criação do *DW*.

Por fim, foi criada uma rotina principal, que executa a chamada dessas duas etapas, de forma a garantir que a identificação do mapeamento seja executada para todos os At_j e passando o devido corpora a ser utilizado por cada etapa. O Algoritmo `mainMapping` apresenta o pseudocódigo dessa rotina.

Os parâmetros passados a essa rotina são os dados transitórios que devem ser integrados ao *DW* (`situational`), o nome dos atributos At_j que foram previamente identificados métricas da tabela fato – e portanto não devem ser considerados no mapeamento – e o que representam (`metrics`), e a porcentagem mínima de votos que as dimensões na primeira etapa do mapeamento devem obter para não serem descartadas (`min_votes`). A saída do algoritmo consiste no mapeamento final obtido.

A porcentagem mínima tem um valor definido por padrão pelo sistema, podendo o usuário redefini-la no momento de realizar a busca, se achar que uma outra porcentagem acarretará em melhor desempenho na etapa de mapeamento. Os dados transitórios e as métricas, por sua vez, são passados pelo processo de DeTEC. Quando esse processo obtém os dados Web, ele realiza uma análise em cima dos mesmos, através de estudo dos metadados, e identifica quais atributos são os objetos de análise requeridos para a consulta do usuário, passando essa informação ao Drill-Conformed. Dessa forma, o operador fica ciente de quais dados não devem ser mapeados em dimensões e devem ser integrados aos dados estacionários como métricas da tabela fato.

Algoritmo mainMapping

Entrada: metrics[], situational[], min_votes
Saída: final_map[]

```
1: //Cada documento contém o domínio de valores de uma das dimensões do DW
2: corpora ← loadDimensionsCorpora()
3:
4: first_map ← getDimensionsMap(metrics, situational, corpora, min_votes)
5:
6: //Para cada At_j mapeado para uma dimensão no primeiro mapeamento
7: //chama o segundo mapeamento para encontrar Ae_i^d
8: for (At_j, De_d) in first_map do
9:    //Cada doc representa o domínio de valores de um atributo da dimensão De_d
10:   corpora ← loadAttrsCorpora(De_d)
11:   if De_d is not a degenerate dimension
12:      Ae_i^d ← getDimensionAttrMap(situational, At_j, corpora)
13:   else
14:      Ae_i^d ← De_d
15:   end if
16:   final_map[De_d][Ae_i^d] ← At_j
17: end for
18:
19: return final_map
```

3.2.5 Componente de Integração

O componente de integração, por sua vez, é responsável por identificar, a partir do conhecimento de qual Ae_i o mapeamento definiu que o atributo At_j deve ser cruzado, a tupla exata para a qual a relação $RI(val(At_j), val(Ae_i))$ é verdadeira e realizar a integração, dando ao usuário novos dados a serem analisados.

Os valores dos atributos de intersecção identificados, apesar de serem dados que em sua maioria condizem com dados existentes no *Data Warehouse*, dificilmente estarão no mesmo formato daquele dos dados estacionários. Por exemplo, um atributo de intersecção denominado `localidade` pode conter um valor no formato "nome_da_rua, nome_da_cidade, estado, pais", enquanto que no *Data Warehouse* tais atributos estão divididos em uma hierarquia e podem não conter todos esses níveis. Ainda, pode ser que o *Data Warehouse* possua os atributos `cidade`, `estado` e `pais` em sua hierarquia, e o atributo de intersecção seja formado por "nome_do_pais, cidade". Dessa forma, tentar realizar a integração a partir da igualdade direta ou combinação com todos os campos não seria uma boa estratégia.

Levando tais pontos em consideração, a lógica adotada visa a tokenização de $val(At_j), \forall At_j \in T$ pertencente à Ci, de maneira que seja possível realizar uma igualdade com os atributos do DW com perda de dados mínima. O algoritmo desenvolvido tem

como abordagem iniciar um `select` a partir do primeiro nível da dimensão que tenha sido mapeado e continuar a busca pelos demais até que reste apenas uma tupla ou as possibilidades tenham sido esgotadas. Antes de explicar o funcionamento do algoritmo, algumas premissas sobre as quais ele foi baseado devem ser apresentadas:

Premissa 4: o $dom(At_j), \forall At_j \in T$, contém, em sua maioria, valores padronizados. Isto é, por se tratarem de atributos qualitativos, geralmente preenchidos por sistemas e não usuários, assume-se que o domínio de valores de um atributo refira-se sempre a um mesmo assunto e de maneira próxima a uma padronização.

Além disso, algumas regras[3] gerais em relação à forma como a integração deve ser conduzida são definidas como:

Regra 1: quando existirem mapeamentos para mais de um atributo Ae_i^d de uma mesma dimensão De_d, deve ser realizada a integração com aquele de menor nível na hierarquia de De_d, uma vez que este sempre traz mais detalhes de informações e opções de navegação por meio dos operadores *OLAP*.

Regra 2: atributos $At_j \in Ci$ para os quais não seja possível encontrar um atributo Ae_i que satisfaça a relação $RI(val(At_j), val(Ae_i))$ devem ter $val(At_j)$ descartado da integração.

Regra 3: atributos $At_j \in Ci$ para os quais existe mais de um Ae_i que satisfaz a relação $RI(val(At_j), val(Ae_i))$ devem ter $val(At_j)$ descartado da integração.

O primeiro passo do algoritmo de integração é identificar a ordem com que os atributos devem ser verificados para cada dimensão contida no mapeamento, quando houver mais de uma. Esse processo é realizado por meio da função `getSortedLevels` (linha 3) e visa preparar os dados para que o cruzamento seja realizado sempre com o nível mais baixo possível da hierarquia (por razões apresentadas na Regra 1), sendo executado da seguinte forma: seja $De_d = \{Ae_1^d, Ae_2^d, Ae_3^d\}$ o conjuntos de atributos $Ae_i \in E$ que fazem parte da dimensão De_d e $H = \{Ae_1^d < Ae_2^d < Ae_3^d\}$ a definição da hierarquia contida em De_d, onde Ae_1^d é o nível mais baixo da mesma e Ae_3^d o mais alto. Esse primeiro passo irá retornar todos os pares $< At_j, Ae_i^d >$, $Ae_i^d \in De_d$, para os quais $\exists At_j \in T | RM(At_j, Ae_i^d)$, ordenados de acordo com a posição de Ae_i^d na hierarquia. Supondo que At_1 e $At_2 \in T$ sejam atributos que satisfazem as relações $RM(At_1, Ae_3^d)$ e $RM(At_2, Ae_1^d)$, ao fim desse passo será retornado o conjunto $\{< At_2, Ae_1^d >, < At_1, Ae_3^d >\}$. Essa é a ordem dos atributos de intersecção que será utilizada para tentar identificar a tupla nos dados estacionários à qual o documento deve ser cruzado na dimensão De_d.

Para montagem da consulta realizada no *Data Warehouse*, a fim de encontrar as possíveis tuplas que satisfazem RI, foram desenvolvidas duas regras gerais, que diferem com base no tipo da dimensão De_d que satisfaz a relação $RM(At_j, De_d)$ para um determinado

[3] Chama-se de regra aquilo que a autora definiu como a norma a ser seguida.

At_j. Supondo que, para um atributo de intersecção At_j, o par $<De_d, Ae_i^d>$ que satisfaz a relação RM seja igual a <D1, A1> e $val(At_j)$ é composto pelos termos individuais T1, T2 e T3, as regras para montagem da consulta que obtém as tuplas iniciais são dadas como segue.

Regra 4: se a dimensão D1 não for temporal, a consulta deverá ser montada de forma a obter todas as tuplas da dimensão D1 cujo atributo A1 é igual a algum dos termos individuais de At_j. Em uma linguagem SQL seria o equivalente a:

```
SELECT * FROM D1 WHERE A1 IN (T1, T2, T3)
```

Regra 5: para o caso em que a dimensão é temporal, a consulta deve ser montada de forma a encontrar um valor equivalente em todos os níveis da hierarquia de D1 a partir do nível A1. Dessa forma, supondo que A1 = mes, a consulta tentaria encontrar tuplas que contêm valores correspondentes aos termos em mês e ano (supondo que a hierarquia da dimensão seja {dia < mes < ano}). Logo, a consulta em uma linguagem SQL seria dada por:

```
SELECT * FROM D1 WHERE mes IN (T1, T2) AND ano IN (T1, T2)
```

Agora que foram definidas as premissas e regras básicas, a lógica de execução do algoritmo de integração proposto pode ser devidamente explicada. Após o algoritmo realizar a ordenação dos atributos associados a cada dimensão (linha 3), o processo realizado para cada documento é o executado como segue: para cada par $<At_j, Ae_i^d>$, para cada dimensão De_d contida no mapeamento, cuja relação $RM(At_j, Ae_i^d)$ é verdadeira, é realizada a tokenização de $val(At_j)$, por intermédio da função `tokenize` (linha 8), que tem como resultado o conjunto de termos individuais de $val(At_j)$. Após a obtenção dos termos, é realizada uma consulta no *Data Warehouse* para obter todas as tuplas da dimensão De_d que poderiam satisfazer a relação $RI(val(At_j), val(Ae_i^d))$, seguindo as regras previamente definidas (linha 6).

A partir dos resultados retornados da consulta, uma de três opções pode ocorrer: (1) nenhuma tupla é retornada; (2) apenas uma tupla é retornada; (3) mais de uma tupla é retornada. No primeiro caso, pelas regras previamente definidas, o $val(At_j)$, para o atributo At_j cujo valor foi tokenizado para essa consulta, deve ser descartado na integração desse documento. O segundo caso significa que a única tupla que pode corresponder a $val(At_j)$ foi encontrada. Assim sendo, $val(At_j)$ do documento é marcado para ser cruzado com essa tupla. Para o terceiro caso, o algoritmo deve continuar sua execução.

Para melhor explicar o que ocorre no terceiro caso, utilizaremos um exemplo: seja $D_{loc} = \{cidade, estado, pais, continente\}$ uma dimensão do DW, cuja hierarquia é

definida como $H_{loc} = \{cidade < estado < pais < continente\}$. Seja $At_{cid} \in T$ o atributo transitório para o qual queremos encontrar o cruzamento e cuja relação $RM(At_{cid}, cidade)$ é satisfeita, sendo $val(At_{cid})$ = "St Louis, North America". Seja os delimitadores utilizados para tokenização de $val(At_j), \forall At_j \in T$ definidos como quaisquer caracteres que não sejam números, letras, espaço, hífen e apóstrofo e, portanto, o resultado da mesma dado por {"St Louis", "North America"}. Suponha que a consulta realizada na linha 6 do algoritmo teve como resultados as tuplas `T1=(`"St Louis", "Missouri", "United States", "North America"`)` e `T2=(`"St Louis", "Saskatchewan", "Canada", "North America"`)`, onde cada campo da tupla representa o valor de um elemento de H_{loc}, na mesma ordem.

Quando mais de uma tupla é encontrada, para cada uma delas, o algoritmo cria um subcaminho, onde cada tupla segue com o seu próprio conjunto de termos obtidos da tokenização de At_j, a partir da eliminação do termo que obteve correspondência com o valor de Ae_i^d para o qual a relação $RM(At_j, Ae_i^d)$ é válida (linhas 12 à 20). No exemplo acima definido, $Ae_i^d = cidade$ e $val(cidade)$ = "St Louis" para ambas as tuplas retornadas. Dessa forma, ambas as tuplas seguem para o seu subcaminho com um conjunto de termos igual a {"North America"}.

O próximo passo do algoritmo é subir um nível na hierarquia da dimensão De_d (linha 9) e, para cada uma das tuplas, verificar se algum dos termos restantes em seu conjunto corresponde com o valor do atributo Ae_i^d nesse nível (linhas 13 à 19). As tuplas que obtiverem correspondência nesse nível continuam e o termo a partir do qual a correspondência é obtida é removido do seu conjunto, enquanto as demais tuplas são eliminadas (linhas 14 à 18). Se nenhuma tupla obtiver correspondência no nível de hierarquia atual, todas são mantidas e nenhuma alteração é feita em seus conjuntos de termos (linhas 21 à 23). Esse processo é repetido até que reste apenas uma tupla, ou o conjunto de termos das tuplas se torne vazio, ou não haja mais níveis na hierarquia para subir (linhas 9 à 25).

No exemplo dado, o atributo Ae_i^d no próximo nível da hierarquia é *estado*, dessa forma, para cada umas das tuplas `T1` e `T2`, verifica-se se $val(estado)$ tem correspondência com "North America". Como nenhuma das tuplas tem correspondência entre o seu conjunto de termos restantes e o valor de seu campo no nível de *estado*, o algoritmo mantém ambas as tuplas e avança para o próximo nível de H_{loc}, sendo este *pais*. Nesse nível, nenhuma correspondência é novamente encontrada para nenhuma das tuplas $T1$ e $T2$ com o conjunto de termos {"North America"}, fazendo com que o algoritmo avance para o próximo nível da hierarquia sem nenhuma alteração nas tuplas ou no conjunto de termos originários do atributo At_{cid}. O próximo nível de H_{ex} é continente. Nesse nível, ambas as tuplas têm correspondência com o termo "North America". Assim sendo, o mesmo é retirado do conjunto de termos de ambas, resultando no conjunto vazio {} tanto para `T1` quanto para `T2` e ambas as tuplas são mantidas. Aqui, o nível final da hierarquia de D_{loc} foi atingido

(além do conjunto de termos de ambas terem se tornado vazios) e ainda existe mais de uma tupla que poderia satisfazer a relação $RI(val(At_{cid}), val(cidade))$, o que nos remete a Regra 3, onde não é possível identificar a tupla de cruzamento e $val(At_{cid})$ deve ser descartado da integração.

Consideremos agora o caso em que existem dois atributos transitórios $At_j \in T$ mapeados para atributos estacionários Ae_i^d distintos, mas que fazem parte da mesma dimensão De_d. Apenas um desses At_j deve ser cruzado com o DW, uma vez que ambos levam à mesma dimensão De_d e, baseando-se na Regra 1, o cruzamento deve ser feito sempre com o nível mais baixo da hierarquia. Porém, o segundo atributo pode ser útil para definir a tupla a ser realizado o cruzamento nos casos em que, como no exemplo acima, não foi possível reduzir os resultados a uma única tupla. Nesse caso, para cada um dos demais pares $< At_j, Ae_i^d >$ contidos no ordenamento realizado no passo inicial do algoritmo para a dimensão De_d (linha 3), todo o processo aqui descrito é repetido, seguindo a ordem em que eles foram ordenados e sempre utilizando as tuplas restantes do processo realizado sobre o par anterior, em vez de realizar um `select` no DW. Esse procedimento é repetido até que reste uma única tupla ou não haja mais pares a serem verificados.

Para explanar esse último caso, vamos estender o exemplo anteriormente apresentado, por meio da adição de mais um At_j, identificado por At_{pais}, que satisfaz a relação $RM(At_{pais}, pais)$ e cujo $val(At_{pais})$ = "United States". Iniciando do fim do processo anterior, onde ambas as tuplas `T1` e `T2` ainda estavam empatadas, o algoritmo define o conjunto inicial de termos, a partir da tokenização de $val(At_{pais})$, como {"United States"}. O nível de H_{loc} a partir do qual a busca será realizada é estabelecido como sendo $pais$ e o processo realizado anteriormente é repetido. Nessa etapa, `T1` tem correspondência no nível de $pais$ com "United States", enquanto que para `T2` não é possível encontrar nenhuma correspondência com o seu conjunto de termos nesse nível da hierarquia. Consequentemente, `T2` e seu conjunto de termos são descartados e `T1` passa para a próxima iteração, tendo como conjunto de termos transitórios o conjunto vazio {}. Uma vez que o algoritmo conseguiu reduzir a quantidade de tuplas a uma, o mesmo é terminado e `T1` é considerada a tupla que satisfaz a relação $RI(At_{cid}, cidade)$ e com a qual o cruzamento deve ser realizado.

O pseudocódigo da rotina aqui descrita é apresentado no algoritmo `integrate`.

Os atributos $At_j \in Cr$ não passam pelo processo de integração, uma vez que não foram identificados como pertencentes a nenhum domínio dos dados estacionários. Porém, ao fim da integração dos atributos de intersecção, esses dados são enviados juntamente com aqueles que foram cruzados, de forma que o usuário possa visualizá-los, se desejar, analisando assim outras informações que possam ser úteis à tomada de decisão.

Algoritmo integrate

Entrada: mapping, metrics[], doc

```
1: //Todos os documentos nos dados transitórios devem ser submetidos a esse processo
2: for De_d in mapping do
3:    attrs ← getSortedLevels(mapping, De_d)
4:    (At_0, Ae_0^d) ← attrs[0]
5:
6:    tuplas ← getFromDW(De_d, Ae_0^d, tokenize(doc[At_0]), getType(De_d))
7:    for (At_j, Ae_i^d) in attrs do
8:       terms ← tokenize(doc[At_j]))
9:       for level in top_dimension_levels do
10:         if len(tuplas) > 1
11:            tuplas1 ← []
12:            for tupla in tuplas do
13:               for remaining_term in tupla do
14:                  if  remaining_term = tupla[level]
15:                     //remove o termo encontrado do conjunto
16:                     removeTerm(tupla, remaining_term)
17:                     tuplas1 ← add(tuplas1, tupla) //Adiciona a tupla ao array tupla1
18:                  end if
19:               end for
20:            end for
21:            if len(tuplas1) > 0
22:               tuplas ← tuplas1
23:            end if
24:         end if
25:      end for
26:   end for
27:
28:   //Se somente uma tupla tiver restado realiza o cruzamento
29:   if len(tuplas) = 1
30:      crossDocument(doc, At_j, De_d, tupla)
31:   end if
32: end for
```

3.2.6 Componente de colaboratividade

Após a realização do mapeamento e integração dos dados, as informações descobertas em relação a estrutura e semântica dos dados transitórios podem ser compartilhadas na Web semântica, a fim de colaborar com a mesma. A etapa de mapeamento obtém informações a respeito do domínio dos dados e o que eles representam, enquanto que a integração encontra relações de equivalência entre esses dados e entre eles e os dados do *DW*. Ambas as informações são úteis para usuários que futuramente lidem com esses domínios de dados.

O componente de colaboratividade é responsável por realizar esse compartilhamento de informações. Uma opção para realização desse processo pode ser por meio da criação

de metadados e utilização de arquivos RDF. A implementação desse componente não é realizada neste trabalho.

3.3 Sumário

Essa capítulo expôs as definições da arquitetura proposta, SelfBI, assim como do novo operador *OLAP* projetado, Drill-Conformed. Tais definições são importantes para entender os objetivos de ambos e a forma como eles visam operar.

A arquitetura proposta visa ser uma arquitetura genérica de integração de dados estacionários com dados transitórios, a fim de permitir a realização de consultas utilizando dados Web e em tempo próximo ao real, dando autonomia ao usuário para realização das mesmas. Tal arquitetura é composta de dois fluxos de dados paralelos, um para os dados transitórios e outro para aqueles estacionários. O foco do trabalho apresentado é o fluxo realizado para os dados transitórios, uma vez que os dados estacionários segue as metodologias utilizadas para *Data Warehouses* e Sistemas de Suporte à Decisão tradicionais.

Dentre as etapas pelas quais os dados transitórios passam desde a extração das suas fontes até estarem prontos para serem visualizados pelo usuário, existem duas que foram o foco principal deste trabalho: o processo de DeTEC (Detecção e extração, Transformação, Enriquecimento e Carga) e a elaboração do operador *OLAP* para manipulação automática dos dados.

O processo de DeTEC tem o intuito de substituir o processo de *ETL* tradicional, adaptando o mesmo para o cenário utilizado e características dos dados transitórios e da arquitetura proposta. Diferentemente da etapa de extração tradicional de uma *ETL*, a etapa de detecção e extração tem o intuito não somente de extrair os dados de suas fontes, mas identificar quais dados, dentre aqueles disponíveis nas fontes previamente definidas, estão realmente relacionados ao assunto buscado pelo usuário. O processo de enriquecimento é do tipo semântico e tem como intuito identificar características dos dados a fim de adicionar conhecimento que possa ser útil à tomada de decisão.

O Drill-Conformed, por sua vez, visa realizar a integração dos dados de maneira autônoma, sem a necessidade de intervenção de um especialista da área de *BI* e/ou TI, fazendo uso somente do domínio dos dados obtidos, e sem acesso aos metadados do mesmo. Isto é, o operador não possui nenhum conhecimento acerca da semântica dos dados, com exceção de algumas informações identificadas pelo processo de DeTEC.

O operador é composto de três componentes: mapeamento, integração e colaboratividade. O algoritmo proposto para o componente de mapeamento é apresentado e baseia-se em um modelo probabilístico e matemático de coocorrência. O algoritmo de integração, por sua vez, utiliza um processo de tokenização dos valores dos atributos e

realiza a comparação um-a-um entre os atributos transitórios e estacionários. Por fim, o componente de colaboratividade é responsável por compartilhar o conhecimento semântico obtido com a Web semântica.

4 Protótipo

Este capítulo visa demonstrar o processo de instanciação da arquitetura SelfBI e o funcionamento do Drill-Conformed quando realizando consultas utilizando dados transitórios. Inicialmente, um estudo de caso é definido, no qual será baseada toda a instanciação dos componentes, de forma a facilitar o entendimento dos processos.

4.1 Estudo de caso: sistema de disponibilização de *streamings*

O cenário escolhido para demonstração da aplicação e funcionamento do trabalho proposto consiste em um sistema de disponibilização de *streamings*, no qual, o gestor, para disponibilizar uma nova *streaming*, necessita comprar os direitos da mesma, de forma que a disponibilização seja feita de maneira legal. A compra desses direitos tem um alto custo e deve ser renovada periodicamente. Em virtude disso, ele precisa selecionar conteúdos que tenham uma boa aceitação com o propósito de: i) causar nos usuários o desejo de assinar seus serviços para assisti-los; ii) manter seu lucro. Além disso, as *streamings* são disponibilizadas por regiões, isto é, com base em diversos fatores, *streamings* podem ou não estar disponíveis em uma determinada região.

Dado o cenário acima, o gestor deseja poder analisar o *feedback* dos usuários nas redes sociais acerca das diversas *streamings* em um dado momento e em diversas regiões, a fim de poder planejar suas aquisições naquele período. Além disso, ele deseja manter dados permanentes no *DW* acerca das *streamings* que disponibiliza, como acessos, o número de sinalizações como favoritas, etc., a fim de ajudar na decisão quanto a renovar ou não os direitos de uma determinada *streaming*.

4.2 Instanciação da Arquitetura

Nesta seção são apresentados os detalhes de desenvolvimento de cada uma das etapas de instanciação da arquitetura para que o operador possa ser utilizado sobre a mesma. Toda a instanciação é feita baseando-se no cenário definido na Seção 4.1.

4.2.1 Fontes de dados

Considerando o cenário de disponibilização de *streamings*, as fontes de dados Web selecionadas foram o Twitter, que na época de seleção das fontes possuía um volume de

dados de aproximadamente 10.600 novos *tweets* por segundo[1], o IMDb[2] e a Netflix. O primeiro foi escolhido como a fonte principal, por disponibilizar um grande volume de opiniões de usuários provenientes de diversos lugares do globo, enquanto que os outros dois foram selecionados como fontes complementares, capazes de prover informações descritivas, como nota, *ranking* e ano de lançamento da *streaming*. Vale ressaltar que, os dois últimos também são úteis como fontes estacionárias, uma vez que também fornecem informações que não são voláteis com tempo, como o ano de lançamento.

A seleção dessas fontes levou em conta, além do volume e diversidade de localização dos dados disponibilizados, a existência de mecanismos para obtenção dos dados, como *APIs*, e a facilidade de uso dos mesmos. Para o Twitter e o IMDb foram utilizadas as *APIs* oficiais dos mesmos, enquanto que para a Netflix foi utilizada, por falta de uma *API* oficial, a Netflix Roullete API[3].

Para os dados estacionários, por sua vez, foram selecionadas fontes capazes de prover informação que simule aquelas que seriam provenientes de um sistema de disponibilização de *streamings*, como: nome, gênero e tipo (filme ou série de TV) de *streamings* e *username* de usuários. Para informações acerca de *streamings*, as fontes selecionadas foram o banco de séries[4] e o The Movie DB[5], que consistem de sistemas Web contendo informações acerca de diversas séries de TV e filmes, respectivamente. Já para *usernames* de usuários, foi também utilizado o Twitter. Além disso, foi utilizado um banco de dados de localidades, Geolite 2[6], para informações acerca das diversas cidades existentes ao redor do mundo.

4.2.2 Armazenamento

Nesta seção são detalhados os modelos de dados que foram definidos e implementados, tanto para os dados estacionários quanto para aqueles transitórios.

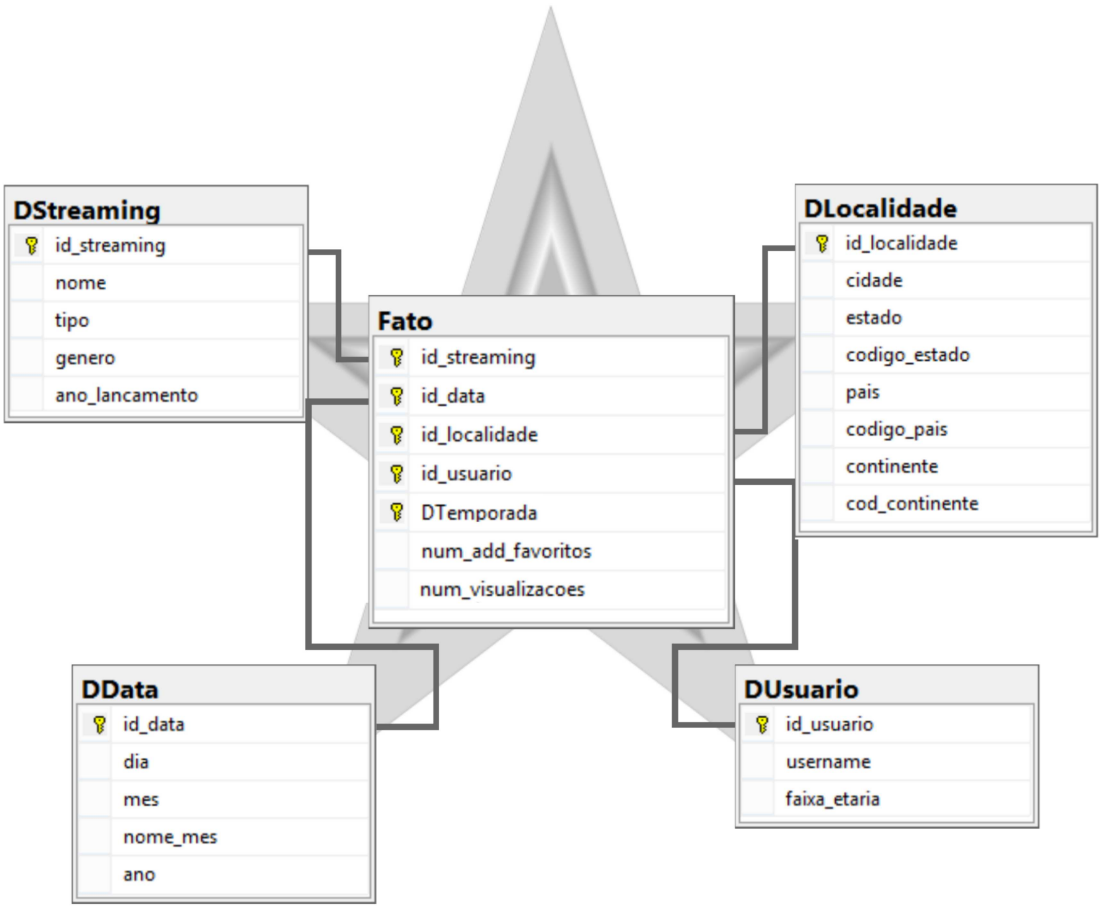
Modelo estacionário

O modelo de dados estacionários foi desenvolvido no SQL Server utilizando a modelagem multidimensional e o esquema estrela. Este modelo é apresentado na Figura 10.

`DData`, `DStreaming`, `DLocalidade` e `DUsuario` são as dimensões básicas, representando, respectivamente, `quando`, `o que`, `onde` e `quem`. Além disso, temos também uma dimensão degenerada (explicada na Seção 2.1.2.1), denominada `DTemporada`. O uso da dimensão degenerada é devido ao fato de que, a única informação descritiva útil referente à

[1] Fonte: *Internet Live Stats*. Acesso em novembro de 2015. Disponível em <http://www.internetlivestats.com>
[2] http://www.imdb.com
[3] http://netflixroulette.net/api
[4] https://bancodeseries.com.br
[5] https://www.themoviedb.org
[6] http://dev.maxmind.com/geoip/geoip2/geolite2

Figura 10: Modelo estacionário



Fonte: Produzido pela autora

temporada de uma *streaming*, quando se tratando de seriados, é a identificação da mesma, a fim de determinar quais temporadas deverão ser disponibilizadas ou retiradas do sistema.

A tabela `Fato`, por sua vez, é responsável pelo armazenamento do objeto de análise, que para os dados estacionários é representado pelo número de acessos que uma *streaming* teve (`num_visualizacoes`) e quantas vezes a mesma foi sinalizada como favorita (`num_add_favoritos`).

O modelo tem como intuito simular os dados que seriam obtidos de um sistema de disponibilização de *streamings* e é capaz de responder consultas como, por exemplo, o número de visualizações (`num_visualizacoes`) da temporada `X` (`DTemporada`), da série `Y` (`DStreaming`) de usuários em uma determinada faixa etária (`DUsuario`) e em um determinado país (`DLocalidade`).

Modelo transitório

O mecanismo de armazenamento escolhido para carga dos dados transitórios foi o MongoDB, que se trata de um sistema de banco de dados NoSQL orientado a documentos. Como já mencionado, sistemas de banco de dados NoSQL são sistemas que não utilizam a modelagem multidimensional e proveem maior flexibilidade em relação a estrutura dos dados a serem armazenados. Sistemas NoSQL orientados a documentos, por sua vez, são aqueles que permitem o armazenamento de estruturas contendo um agregado de dados, geralmente relacionadas, denominadas documentos. XML e JSON são bons exemplos destes (SADALAGE; FOWLER, 2013). A escolha de tal mecanismo se deu pela simplicidade e rapidez de inserção providas pelo NoSQL e em razão de poder armazenar documentos em qualquer estrutura. Tais características são benéficas para o nosso propósito, uma vez que lidamos com dados Web, em formato JSON, e o operador proposto tem como premissa o não conhecimento da estrutura dos mesmos.

Por decisão de desenvolvimento, um segundo mecanismo de armazenamento de dados transitórios foi adicionado, com o intuito de tirar proveito do servidor *OLAP*. Esse armazenamento auxiliar e temporário é realizado após o operador realizar o mapeamento e integração dos dados, para processar as consultas do usuário. Para simplificar a implementação, criou-se um modelo unificado dos dados transitórios e estacionários, a fim de responder as consultas realizadas com ambos os dados. Assim sendo, os dados transitórios, após a integração e até o fim do seu ciclo de vida dentro do sistema, também são armazenados no SQL Server, como uma outra tabela fato, denominada "Fato transitório". Essa tabela está ligada as mesmas dimensões da `Fato` original e contém como objeto de análise a nota média da *streaming* e o número de *feedbacks* positivos, negativos e neutros que a mesma obteve a partir dos *tweets* obtidos. Dessa forma, o esquema apresentado na Figura 10 se torna um esquema constelação, representado na Figura 11.

Ainda, vale ressaltar que, para o desenvolvimento deste trabalho, todos os dados utilizados, sejam eles estacionários ou transitórios, se encontram na língua inglesa, por motivos de ser um idioma que possui um volume de dados mais abrangente.

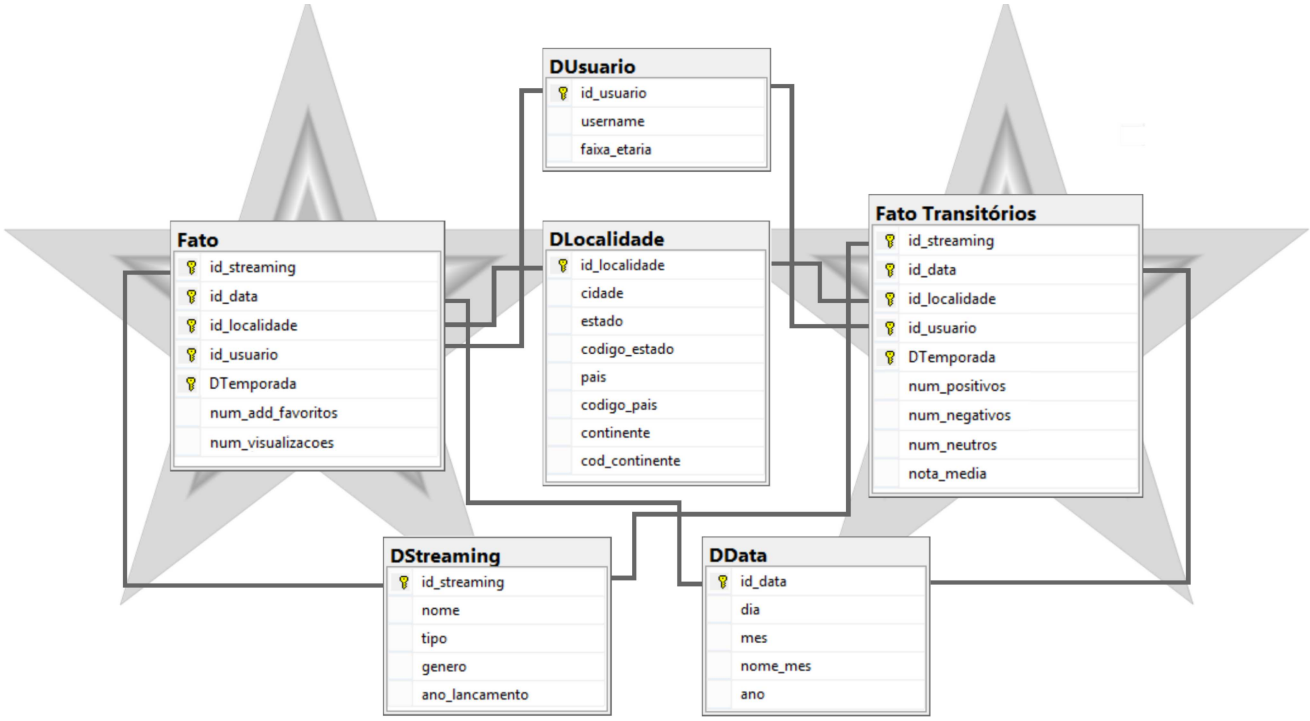
4.2.3 Processos de *ETL*

Assim como para os mecanismos de armazenamento, também é necessário criar processos de *ETL* para lidarem com os dados estacionários e transitórios de maneira distinta, o que resultou na criação de dois processos separados.

4.2.3.1 *ETL* Dados Estacionários

Como mencionado anteriormente, o processo de *ETL* dos dados estacionários segue os passos de uma *ETL* tradicional. As etapas para criação do mesmo podem ser definidas

Figura 11: Modelo de integração - Dados transitórios e estacionários



Fonte: Produzido pela autora

em duas fases principais, extração dos dados de suas fontes e pré-processamento dos mesmos.

Extração

Com base no cenário escolhido, para as fontes de dados estacionários consistindo puramente de sistemas Web – Banco de séries, The movie DB –, foi desenvolvido um *web crawler*, que percorre todas as páginas dos mesmos que contém dados relacionados às informações úteis ao nosso modelo estacionário (apresentado na Figura 10). As informações extraídas a partir desse mecanismo foram nome, gênero, tipo (filme ou série de TV) e ano de lançamento. Ao todo foram extraídos dados referentes a 12.061 seriados e 18.641 filmes, totalizando 30.702 *streamings*.

Já para a dimensão `DLocalidade`, que armazena informações acerca de cidades existentes ao redor do mundo, foi feito o *download* do banco de dados disponível no Geolite 2. Ao todo foram extraídas informações referentes a 72.500 cidades. Para as informações temporais, foram geradas todas as datas possíveis que abrangiam o período de desenvolvimento deste trabalho e dos dados nele utilizados. As informações geradas

foram dia, número do mês no ano, nome por extenso do mês (no formato abreviado em inglês) e ano.

Para a dimensão `DUsuario` foi utilizada a Streaming API do Twitter[7]. Foi criada uma rotina que rodou continuamente durante aproximadamente 6 horas, obtendo novos *tweets* da *API*. Os *usernames* dos usuários relacionados a esses *tweets* obtidos foram então armazenados em um sistema de banco de dados, da onde foi realizada a extração para o *Data Warehouse*.

Por fim, para as demais informações nas dimensões – faixa etária para a dimensão `DUsuario` e o número de temporada ao qual consiste a dimensão `DTemporada` – foram gerados dados a partir de algoritmos pseudoaleatórios, pois os mesmo não estavam disponíveis nas fontes utilizadas. Para o campo `faixa_etária` foram considerados quatro tipos de faixa etária (infantil, jovem, adulto e idoso), que foram atribuídos aleatoriamente a cada um dos nomes de usuários extraídos anteriormente. A dimensão `DTemporada`, por sua vez, recebeu valores no intervalo de 1 à 100, representando o número da temporada do seriado[8]. Para os casos em que a *streaming* é um filme, a dimensão de temporada não se aplica.

Os dados a serem inseridos na tabela fato também foram obtidos a partir de valores aleatórios. Um algoritmo foi desenvolvido para obter uma combinação de chaves primárias das dimensões e em função destas construir a tupla a ser inserida na `Fato`.

Pré-processamento

O pré-processamento dos dados estacionários também foi realizado seguindo metodologias tradicionais, comumente aplicadas na criação de *Data Warehouses*.

Para os dados inseridos na dimensão `DStreaming`, após a extração dos mesmos de suas fontes, foram removidos caracteres especiais e acentuação dos nomes das *streamings* e seu gênero. O mesmo para o campo `username` da dimensão `DUsuario`. Para aquelas *streamings* cujo gênero não pôde ser obtido, por não conter tal informação no sistema Web utilizado, o valor "*unknown*" foi adicionado no campo. Após esse processamento, foram verificadas a existência de *streamings* duplicadas, – mesmo nome, ano de lançamento e tipo – sendo tais tuplas removidas, deixando apenas uma tupla referente à *streaming* e realizando a junção de seus gêneros quando os mesmos divergiam, tomando o devido cuidado para não duplicá-los. Para os *usernames* obtidos através do Twitter, também foram removidas duplicatas, deixando um total de 47.399 *usernames*.

[7] https://dev.twitter.com/streaming/overview

[8] É de conhecimento da autora de que não existem, atualmente, seriados de TV com 100 temporadas. Sendo Doctor Who o de maior duração até o momento de escrita desta dissertação, composto de 36 temporadas. O número elevado foi escolhido apenas no intuito de ter um maior domínio de dados.

Em relação aos dados da dimensão `DLocalidade`, após a obtenção do banco de dados foi feita uma análise da estrutura das informações contidas no mesmo e seus domínios, a fim de identificar o que cada campo representava. Uma vez que a análise foi devidamente realizada, um *script* foi criado para transformar as informações obtidas em instruções `insert` do SQL Server compatíveis com a estrutura da dimensão, estas que foram posteriormente executadas sobre o *DW*. Para as localidades cujo nome de estado não se aplica, em função do país não ser dividido em estados ou alguma divisão semelhante, o valor "*Not Applicable*" foi inserido no campo `estado` e "NA" no campo `codigo_estado`.

Para os demais campos e dimensões, por se tratar principalmente de informações propriamente geradas para desenvolvimento deste projeto, nenhum pré-processamento se fez necessário.

4.2.3.2 DeTEC

Todo o desenvolvimento do processo de DeTEC foi realizado através da criação de *Web Services RESTful*, utilizando as linguagens de programação C# e Python.

Detecção e extração

A etapa de detecção instanciada consiste em montar uma consulta interna [9] e enviá-la para as *APIs* sendo utilizadas, de forma que, em sua maioria, somente dados referentes à *streaming* sejam detectados. Enquanto que o passo de extração é encarregado de extrair de fato os dados detectados de suas fontes para o sistema. Para isso, foi criado um *endpoint* que recebe do usuário, como parâmetros, o nome da *streaming* a ser buscada e o seu tipo (filme ou TV Show).

Para a parte de detecção, foi desenvolvido um algoritmo que monta uma consulta interna que é enviada à *REST API* do Twitter. Tal *API* provê uma série de operadores a serem utilizados na montagem da consulta a ser realizada sobre a mesma[10]. Para o algoritmo criado, os seguintes operadores foram utilizados:

- **OR:** utilizado no formato `valor1 OR valor2`. Encontra *tweets* que contenham o `valor1` ou o `valor2`, ou ambos.
- **Hashtag(#):** utilizado no formato #`valor`. Detecta *tweets* que contenham *hashtags* com o valor passado.
- **Operador padrão:** funciona como um `AND`, retornando *tweets* que contenham o `valor1` e o `valor2`. Para utilização desse operador basta colocar os dois valores

[9] Neste trabalho, chamam-se de consulta interna aquelas executas pelo sistema; e de consulta externa (ou somente consulta) aquelas realizadas pelo usuário.

[10] Fonte: *Twitter Developer Documentation*. Acesso em abril de 2017. Disponível em <https://dev.twitter.com/rest/public/search>

de maneira adjacente separados por um espaço e sem nenhum operador entre eles. Ex: uma consulta formada por "Olá mundo" (sem passar as aspas na consulta) irá retornar apenas *tweets* que contenham as palavras "Olá" e "mundo" simultaneamente.

O algoritmo de criação da consulta interna consiste na junção dos parâmetros passados, utilizando os operadores acima citados, com algumas palavras chaves e *hashtags* geradas com base no tipo da *streaming* sendo buscada e no número de palavras contidas na mesma. Suscintamente, o algoritmo gera duas expressões da consulta de maneira separada, uma com variações do nome da *streaming* e outra com possíveis combinações do tipo da mesma, utilizando o operador `OR` e duplicando alguns dos valores contidos em cada expressão na forma de *hashtag*. Posteriormente, as duas expressões são unidas por intermédio do operador padrão (`AND`), a fim de obter *tweets* que possuam alguma variação do nome da *streaming* juntamente com alguma outra palavra chave que indique se tratar de fato do filme ou seriado de TV e não um outro assunto qualquer. O Algoritmo `BuildQuery` apresenta o pseudocódigo do algoritmo criado.

Como pode ser visto, as variações utilizadas para o nome da *streaming* são o nome sem delimitadores (espaço, hífen, dois pontos etc) e as iniciais da mesma, quando contendo mais de duas palavras[11], ambas representadas no formato de *hashtag*. Por exemplo, para a série de TV "The Good Wife", as variações seriam #TGW e #TheGoodWife. O uso de *hashtag* aqui se deu pelo fato de ser muito comum a utilização das mesmas em textos de *microblogs*. Do mesmo modo, a remoção de espaços e as siglas são muito comuns quando na sua utilização, além de que seriados e filmes são frequentemente referenciados por suas siglas nesses mecanismos, quando possuem nomes muito grandes, devido, principalmente, a limitação de caracteres existente nos mesmos. Na linha 27 do algoritmo, é realizada a concatenação dessas informações através do operador `OR`, resultando na primeira expressão da consulta.

A função `getCombinations` (executada na linha 20), por sua vez, é responsável por obter um conjunto de combinações do tipo da *streaming*, que serão posteriormente concatenadas usando o operador `OR`, formando a segunda expressão da consulta. Para *streamings* do tipo filme, o resultado obtido é um *array* de quatro posições, contendo os valores `movie`, `#movie`, `movies` e `#movies`. Para *streamings* do tipo série de TV é gerado um número maior de combinações, apresentadas na Tabela 3.

Por fim, ambas as expressões geradas são concatenadas mediante utilização do operador padrão (linha 30 do algoritmo), resultando na consulta final a ser enviada para a *API*.

[11] Não são utilizadas as iniciais para *streamings* de duas palavras porque existe uma vasta quantidade de siglas e abreviações no vocabulário da internet que utilizam duas letras apenas, o que diminuía a precisão do algoritmo.

Algoritmo BuildQuery

Entrada: streaming_name, streaming_type
Saída: a url da consulta

```
1:
2: API_URL ← url do endpoint a ser usado
3: name_no_delimiters ← removeDelimiters(streaming_name)
4: //tokeniza o nome da streaming em palavras individuais
5: name_words ← getWords(streaming_name)
6:
7: //Obtém as iniciais do nome da streaming
8: if len(name_words) > 1
9:    if len(name_words) > 2
10:      abbreviation ← getAbbreviation(name_words)
11:   else
12:      abbreviation ← name_no_delimiters
13:   end if
14: else
15:   abbreviation ← streaming_name
16: end if
17: //Obtém as combinações possíveis do tipo da streaming, incluindo as hashtags
18: combinations ← getCombinations(streaming_type)
19: //Obtém as Hashtags da abreviação e do nome sem espaços da streaming
20: no_delimiters_hash ← getHashtag(name_no_delimiters)
21: abbreviation_hash ← getHashtag(abbreviation)
22:
23: //Concatena os parâmetros passados utilizando o operador OR
24: typeOr ← concatOR(streaming_name, no_delimiters_hash, abbreviation_hash)
25: nameOR ← concatOR(combinations)
26:
27: //Concatena os parâmetros passados utilizando o operador padrão (AND)
28: query ← concatAND(typeOR, nameOR)
29:
30: return  concat(URL_API, query)
```

A etapa de extração, por sua vez, é responsável por conseguir obter todos os *tweets* que foram encontrados na etapa de detecção. A cada requisição feita à *API* do Twitter é retornado um máximo de 100 *tweets*, juntamente com uma variável que informa a *URL* dos próximos *tweets*, se houverem. Desta forma, foi criado um algoritmo que inicia com a *URL* criada pelo processo de detecção e executa em *looping* até que nenhuma *URL* seja retornada no campo de próximos resultados. Esse processo é feito sempre atualizando a *URL* para aquela retornada na resposta a cada iteração. Uma vez que todos os *tweets* foram obtidos o processo de transformação dos mesmos é acionado.

Tabela 3: Conjunto de combinações - Série de TV

Combinação	**Significado**
"tv show"	Contém a expressão “tv show”
#tv#show	Contém as *hashtags* “tv” e “show”, não necessariamente adjacentes
#tvshow	Contém a *hashtag* “tvshow”
#tvserie	Contém a *hashtag* “tvserie”
season	Contém as palavras “season”
#season	Contém a *hashtag* “season”
episode	Contém a palavra “episode”
#episode	Contém a *hashtag* “episode”

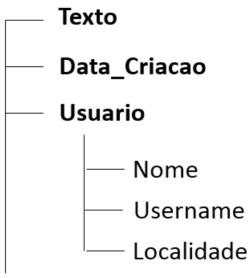
Transformação

Para implementação do processo em função do estudo de caso ao qual a arquitetura está sendo instanciada, quatro transformações foram realizadas. Como dito anteriormente, a adição de novas transformações e/ou tratamento dos dados obtidos são identificados através das informações obtidas da Web semântica, uma vez que não se tem conhecimento prévio dos dados. Porém, para desenvolvimento do protótipo aqui apresentado, essas transformações foram definidas de forma manual pela autora. As duas principias transformações estão relacionadas a limpeza e reestruturação dos dados, a terceira é consequência de uma deficiência presente na *API* do Twitter no momento de desenvolvimento deste projeto e a última está relacionada ao cenário.

As duas primeiras transformações têm como intuito reestruturar os JSONs obtidos a fim de facilitar o trabalho a ser realizado pelo o operador proposto. A primeira consiste em remover os atributos do tipo aninhado dos *tweets* retornados, transformando cada um dos seus campos em atributos que residem na raiz do JSON. Os nomes dos novos atributos obtidos a partir desse processo são construídos por meio da concatenação do nome do atributo aninhado que foi eliminado com o nome do campo que foi trazido para a raiz. As Figuras 12 e 13 ilustram a estrutura de um JSON antes e depois dessa transformação, respectivamente. A segunda transformação, por sua vez, resume-se em remover dos *tweets* obtidos atributos cujo rótulo indica que o mesmo é um ID, isto porque, tais atributos são compostos de uma cadeia numérica de identificação que não possui nenhum significado para tomada de decisão.

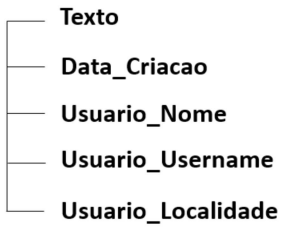
Atualmente, a *API* do Twitter está com problemas em informar a localização de postagem do *tweet*. O atributo dos *tweets* retornados que deveria conter esse valor, identificado como `coordinates`, está retornando apenas valores nulos. Por definição, esse campo deveria retornar um *array* contendo os valores de longitude e latitude. Dessa forma, a primeira transformação realizada consiste em preencher os valores de localização dos resultados retornados. Para isso, foi criada uma rotina que, para cada *tweet* obtido, gera

Figura 12: JSON antes da transformação



Fonte: Produzido pela autora

Figura 13: JSON depois da transformação



Fonte: Produzido pela autora

valores aleatórios de longitude e latitude (dentro dos devidos intervalos), preenchendo o campo com as devidas coordenadas. Para obtenção das localizações de maneira legível, foi utilizada a *API* de *geocoding* do Google Maps[12], por intermédio do método de geocodificação inversa.

Por fim, a última transformação tem como intuito obter o ano de lançamento e a nota atribuída à *streaming* buscada. Tanto a *API* do The Movie DB quanto aquela da Netflix retornam uma nota para a *streaming*, quando a mesma consta em seus sistemas, e o ano de lançamento da mesma. Consequentemente, é necessário fazer a junção desses dados a fim de obter um valor único de cada um. O método adotado para obter a nota é por meio da média aritmética das duas e, no caso em que apenas uma nota é obtida, essa é a nota armazenada.

Enriquecimento

No cenário sobre o qual este projeto é aplicado, é adicionada uma nova métrica, representada pela adição do campo `sentiment` a cada *tweet*. Nesse novo campo é armazenado o resultado da análise de sentimento feita sobre o texto do *tweet*, a fim de identificar a opinião do usuário acerca da *streaming* buscada. Para isso, o processo deve ser capaz de identificar o campo que representa o texto sobre o qual a análise de sentimento deve ser realizada. Como apresentado na Seção 3.1.3.1, tal identificação é feita através da análise das informações extraídas da Web semântica, como metadados. No entanto, pelos mesmos motivos apresentados na etapa de transformação, para o protótipo desenvolvido, essa informação, assim como os demais campos adicionados abaixo, foi configurada manualmente.

Para obtenção da análise de sentimento foi utilizada uma *API* externa. A escolha da *API* utilizada se deu a partir do estudo de algumas *APIs* de análise de sentimento conhecidas. As principais questões que foram levadas em consideração na escolha foram o

[12] https://developers.google.com/maps/documentation/geocoding/start

tempo de processamento por parte da *API* e se a mesma é de uso gratuito ou não. Ao fim, a *API* escolhida foi a sentiment140[13].

Além disso, dois outros novos campos são adicionados a cada um dos *tweets* obtidos, sendo eles, o nome e tipo da *streaming* sendo buscada. O valor de ambos os campos são aqueles informados pelo usuário no momento da busca.

Carga

A etapa de carga do processo de DeTEC carrega os dados extraídos da Web no mecanismo de armazenamento previamente definido. Esse mecanismo foi definido como sendo o MongoDB, como apontado na Seção 3.1.3.1. Dessa forma, os JSONs obtidos e manipulados nas etapas anteriores podem ser diretamente carregados no banco, sem necessidade de adaptações a esquemas e mantendo assim suas estruturas. No intuito de facilitar o processamento das consultas, um único documento é armazenado no MongoDB, contendo o nome e tipo da *streaming*, o UUID único (gerado pelo próprio sistema de banco de dados) e um *array* com todos os *tweets*. Somente nos casos em que há muitos *tweets* e o documento se torna demasiado grande para ser armazenado que é realizada a divisão em mais de um documento.

4.3 Drill-Conformed: Instanciação

Em virtude do operador ter o intuito de ser um componente genérico, que possa ser instanciado em diversos cenários, a implementação realizada foi aquela apresentada na Seção 3.2, sem adaptações, a fim de analisar se é possível utilizar o mesmo com um certo nível de generalidade.

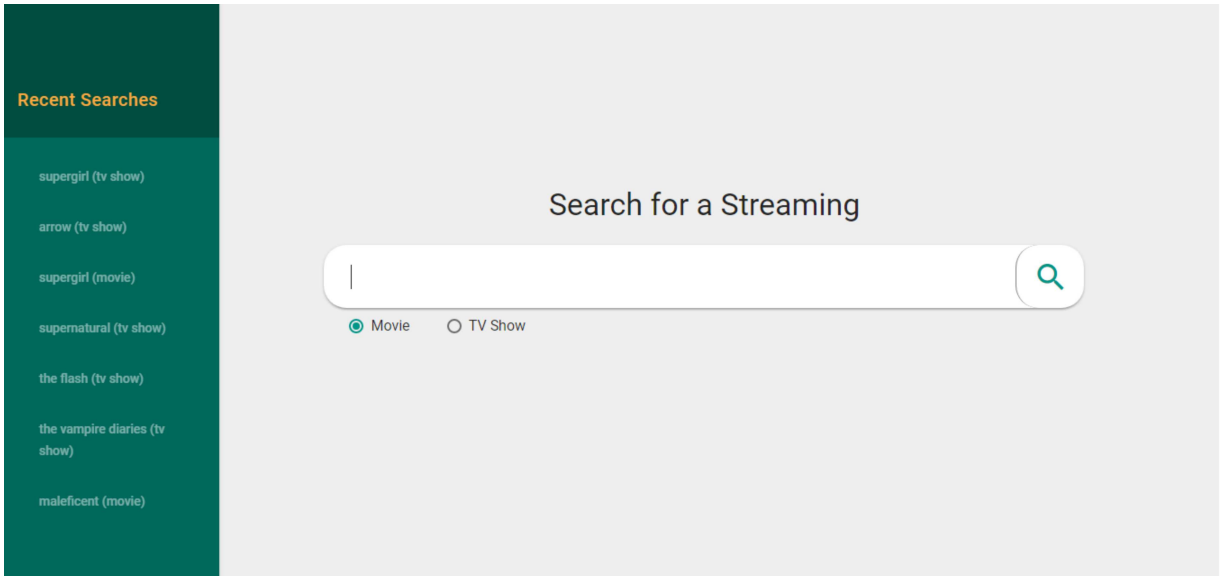
Com este fim, um sistema Web foi implementado. O sistema oferece uma interface onde o usuário pode realizar as consultas. Sendo o foco os dados transitórios, a interface prevê consultas envolvendo esses dados. O sistema instancia a arquitetura proposta e, por conseguinte, o operador Drill-Conformed.

A primeira etapa do sistema consiste na inicialização da busca por parte do usuário, a partir da inserção de uma *streaming* a ser buscada (Figura 14). Além do nome da *streaming* o usuário deve selecionar o tipo da mesma: filme ou seriado de TV.

Uma vez que o usuário informou os parâmetros necessários, uma tela de diálogo é aberta, pedindo que ele informe a porcentagem de confiabilidade com a qual ele deseja que o mapeamento trabalhe. Como explicado na Seção 3.2.4, na primeira etapa do mapeamento o operador considera uma porcentagem mínima de votos para julgar válido o mapeamento de um dado atributo. O valor mínimo padrão para essa porcentagem foi definido a partir

[13] http://help.sentiment140.com/api

Figura 14: Protótipo - Tela de busca



Fonte: Retirada do protótipo desenvolvido pela autora

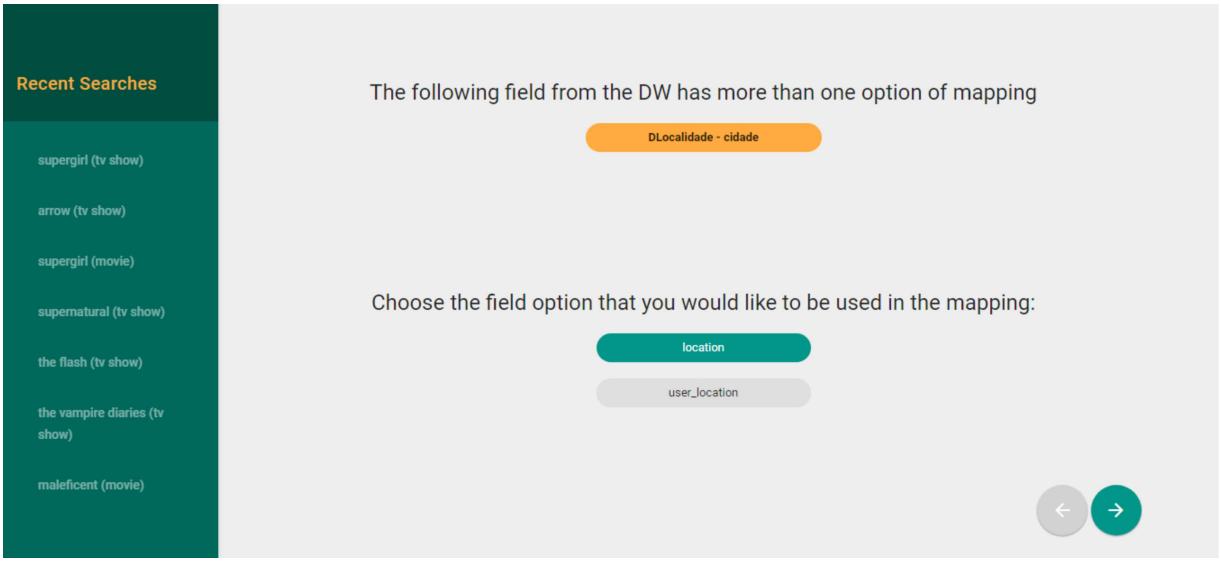
de uma busca em *grid* realizada, variando a porcentagem entre 10% e 80%. O valor final escolhido foi o de 60%, mas o usuário é livre para modificá-lo antes da execução do operador, caso acredite que um valor diferente retornará resultados mais satisfatórios ao seu propósito. Os resultados da experimentação para cada valor testado são apresentados no Capítulo 5.

Após a porcentagem de votos ser definida pelo usuário, inicia-se o processo de DeTEC. Enquanto esse procedimento é realizado, o que pode levar alguns minutos, uma mensagem é mostrada na tela para o usuário, de forma que ele saiba que o processamento está sendo realizado. Quando o processo de DeTEC finaliza seu trabalho, o operador é acionado e o processo de mapeamento se inicia, mudando a mensagem que é mostrada na tela para uma que informe que o estado atual é o de mapeamento dos dados.

Ao fim do mapeamento, caso haja campos do *Data Warehouse* que foram mapeados para mais de um atributo dos *tweets* obtidos, para todos os campos que se encaixam nessa condição, é mostrada uma tela para que o usuário escolha, dentre os atributos transitórios que foram mapeados, com qual ele deseja que o cruzamento seja de fato realizado (Figura 15). Nos casos em que o *DW* possui dimensões com múltiplos papéis, essa configuração pode ser modificada para representar dimensões distintas representadas pelo nome de seus papéis, por exemplo.

Para finalizar a tarefa inicial do operador, é mostrado ao usuário o mapeamento final obtido (Figura 16). O usuário pode modificá-lo caso deseje. A etapa de integração

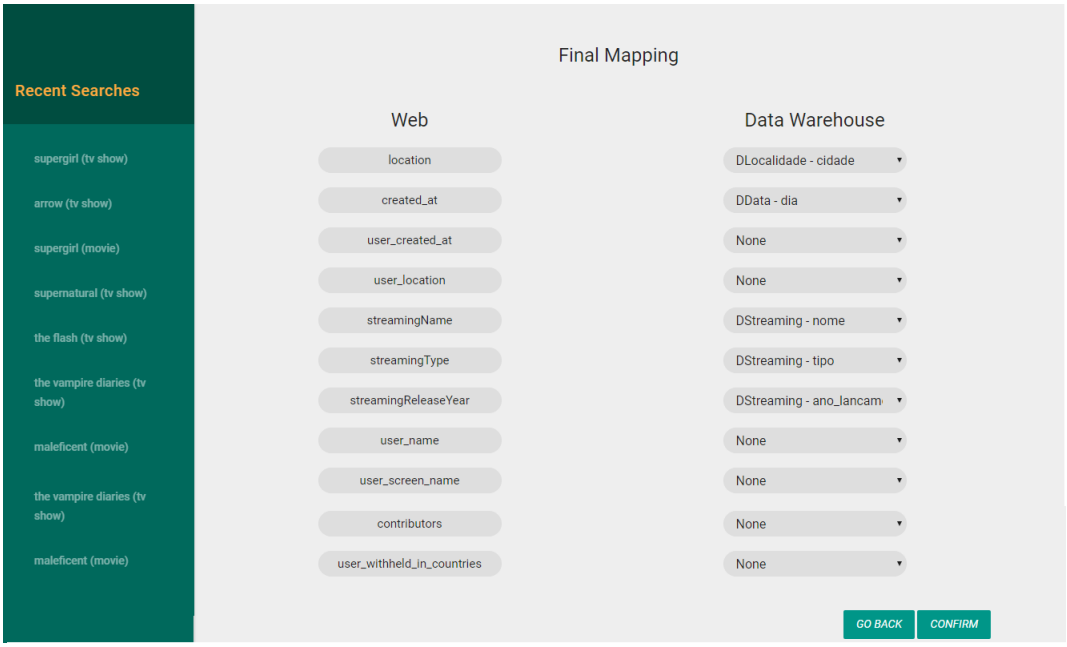
Figura 15: Protótipo - Múltiplas opções



Fonte: Retirada do protótipo desenvolvido pela autora

se inicia, também exibindo uma mensagem na tela para o usuário enquanto o processo é realizado. Uma vez que a integração é finalizada, o usuário é notificado e pode visualizar os resultados da consulta executada e navegar sobre os dados.

Figura 16: Protótipo - Mapeamento final



Fonte: Retirada do protótipo desenvolvido pela autora

4.4 Sumário

Nesse capítulo foi apresentado o protótipo criado para demonstrar o funcionamento do operador proposto e o fluxo de informações da arquitetura quando realizam-se consultas envolvendo dados transitórios.

Os detalhes de implementação aplicados para cada uma das etapas da arquitetura proposta foram apresentados, tanto para dados estacionários como transitórios, além daqueles relacionados ao operador Drill-Conformed. Além disso, um sistema Web foi desenvolvido para exibir o funcionamento das etapas descritas na Figura 8 do Capítulo 3.

Apesar de o operador ter o intuito de ser autônomo, o usuário é capaz de interagir com o mesmo na interface durante algumas etapas do processo, podendo fazer modificações nos resultados parciais apresentados para ele, a fim de melhorar a sua experiência e/ou o resultado final para a tomada de decisão.

5 Avaliação experimental

O presente capítulo expõe a avaliação experimental realizada com o intuito de validar o processo de *ETL* dos dados transitórios dentro do contexto da arquitetura e o operador propostos, tanto em termos de desempenho semântico como em tempo de processamento. Inicialmente, são apresentadas as configurações utilizadas para realização dos experimentos, detalhando as métricas utilizadas e a definição de parâmetros. Na sequência, são apresentados os resultados obtidos em relação ao processo de DeTEC e à cada etapa do operador Drill-Conformed, mapeamento e integração.

5.1 Configuração dos experimentos

Nessa seção é especificada a metodologia e outros parâmetros estabelecidos para a execução dos experimentos aplicados, com o intuito de validar o protótipo da arquitetura apresentada e o operador *Drill-Conformed* proposto.

5.1.1 Metodologia

Os experimentos foram realizados utilizando um conjunto de *tweets* como base dos dados transitórios. Tais *tweets* estavam relacionados a 16 *streamings*, entre filmes e seriados de TV, em um total de 74.015 *tweets*, que foram obtidos a partir de consultas realizadas à *API* do Twitter. No que diz respeito aos dados estacionários, o *Data Warehouse* utilizado possui um tamanho de 81MB, correspondendo a um total de 338.781 tuplas, entre dimensões e tabela fato.

Os processos foram divididos em três experimentos – DeTEC, componente de mapeamento e componente de integração – e cada um deles foi submetido a dois tipos de avaliação experimental, uma em relação ao seu desempenho em termos de semântica e a outra acerca do tempo de processamento. A metodologia utilizada para cada um deles é definida a seguir.

Experimento I - DeTEC

Para avaliação experimental em relação ao desempenho semântico do algoritmo criado para detecção e extração dos *tweets*, foi utilizado um método de validação manual, de forma que dois usuários classificaram, em contraste com a *streaming* sendo buscada, cada um dos *tweets* retornados com um de três valores possíveis, expostos na Tabela 4. Ambos os usuários classificaram cada um dos *tweets* adquiridos. Ao fim, para eliminar

discordâncias, aqueles *tweets* em que as opiniões dos usuários divergiram foram classificados como "não identificado".

Tabela 4: Possíveis valores de classificação do *tweet*

Valor	**Descrição**
relacionado	O *tweet* de fato diz respeito a *streaming* buscada
não relacionado	O assunto do *tweet* não está relacionado à *streaming* buscada
não identificado	Não foi possível identificar se o *tweet* está relacionado à *streaming*

Para facilitar a análise e identificação de pontos críticos do algoritmo, as *streamings* foram manualmente classificadas em sete categorias, de acordo com características do seus títulos em inglês. A Tabela 5 exibe as categorias extraídas.

Tabela 5: Categorias das *streamings*

Categoria	**Descrição**	***Streamings***
1	Contém uma única palavra e a mesma é muito comum e pode ser utilizada em inúmeros contextos e de maneiras diversas	Friends, Saw
2	Contém uma única palavra e esta é comum	Castle, Lucifer
3	Contém uma única palavra que não é frequentemente usada na língua	Maleficent, Shadowhunters, Supernatural, Zootopia
4	Composta de duas palavras	Teen Wolf, The Flash, Modern Family
5	Título de contexto específico/fictício, como histórias em quadrinhos e super-heróis	Capitain America: Civil War, X-men: Apocalipse
6	Título grande, com três palavras ou mais	How I Met Your Mother, The Good Wife
7	Título grande que representa expressões do cotidiano e pode ser usado em outros contextos	Friends with benefits

Devido ao alto volume de *tweets* coletados, uma amostragem foi extraída e a classificação manual foi realizada sobre ela. Para cada *streaming*, uma amostra do seu conjunto de *tweets* obtidos foi utilizada, contendo no máximo 100 elementos. A escolha dos elementos das amostras foi feita de maneira aleatória, e a cardinalidade da amostra extraída para cada *streaming* foi definida a partir das seguintes regras:

1. Se o número total de *tweets* não for maior que 75, todos os *tweets* são utilizados;
2. Se 30% dos *tweets* não ultrapassa a quantidade de 100 *tweets*, 30% é utilizado;

3. Se 10% dos *tweets* não ultrapassa a quantidade de 100 *tweets*, 10% é utilizado;

4. 100 *tweets* são utilizados.

Ao todo, 2.186 *tweets* foram classificados e utilizados para avaliar o desempenho semântico do processo.

Para o tempo de processamento, dois processos são verificados: i) o tempo requerido para detecção e extração dos *tweets*; ii) o tempo de processamento da *API* de análise de sentimento utilizada no desenvolvimento deste trabalho. Os processos escolhidos são os pontos de gargalo do processo de DeTEC desenvolvido. Para obtenção do resultado final, o algoritmo é executado três vezes e o tempo médio é utilizado.

Experimento II - Componente de mapeamento

A validação experimental do operador proposto é feita em duas etapas, uma para analisar o desempenho do componente de mapeamento e outra para o componente de integração. Em ambos os casos foram examinados o desempenho semântico e tempo de processamento dos componentes.

Para o componente de mapeamento, inicialmente é realizada uma busca em *grid*, a fim de identificar o melhor valor para o parâmetro que define a porcentagem mínima de votos que uma dimensão dos dados estacionários deve ter para que seu mapeamento seja considerado válido, o qual chamaremos de `perc`. Posteriormente, são realizadas as duas validações sobre o mesmo, aquela semântica, que se refere a corretude do mapeamento obtido, e aquela sobre o tempo de processamento, uma vez que um dos propósitos do trabalho consiste em realizar o processo em tempo próximo ao real.

Os valores de `perc` utilizados na busca em *grid* são aqueles contidos no intervalo [10, 80] com passo 10. O número de *tweets* utilizados para realizar esse procedimento foram 10.000 e 2.000, que são, respectivamente, a média de *tweets* retornados em buscas acerca de *streamings* que tiveram foco muito recentemente e aquelas que estão fora dos holofotes há um tempo.

Após estabelecer a porcentagem mínima de votos, o mapeamento é executado sobre alguns *datasets* de JSONs retirados da Web[1,2], a fim de validar o seu desempenho semântico em um conjunto de dados mais abrangente e não somente sobre a estrutura dos *tweets*. O critério de escolha dos *datasets* foi a existência de atributos que pudessem ser mapeados no modelo de *DW* criado para o estudo de caso sobre o qual este projeto foi desenvolvido. A Tabela 6 apresenta os *datasets* que foram utilizados e suas características.

[1] http://jsonstudio.com/resources/
[2] https://catalog.data.gov/dataset?res_format=JSON

Tabela 6: JSON Datasets

Nome	Descrição	# documentos
Aeroportos	Informações acerca de diversos aeroportos do mundo	3.885
Zip Codes	Informações de cidade e código postal de locais nos EUA	29.467
Bancos	Informações acerca de bancos ao redor do mundo	500

Para avaliação do tempo de processamento, o componente é executado sobre conjuntos de *tweets* com cardinalidades diversas e o mesmo é medido em segundos.

Experimento III - Componente de integração

Para a validação semântica do componente de integração, foi necessário identificar, manualmente, para cada par <atributo, tupla> integrado, se o valor pertencente ao atributo do *tweet* realmente correspondia ao valor da tupla. Além disso, também foi verificado se os valores que não foram integrados a nenhuma tupla realmente não tinham equivalente dentro do *Data Warehouse*.

Para um conjunto de 10.000 *tweets*, onde 4 atributos obtiveram um mapeamento para campos do *Data Warehouse*, o total de valores envolvidos na integração é igual a 40.000, o que torna infactível a verificação manual de todos os valores do conjunto. Assim sendo, foram extraídas três amostras dentre os *tweets* obtidos, sobre as quais a validação semântica do componente foi realizada. Tais amostras são compostas de 200 *tweets* cada, selecionados de maneira aleatória e sem intersecção entre seus conjuntos de elementos. A Tabela 7 exibe as características de cada amostra extraída.

Tabela 7: Amostras

Nome	# *tweets*	# valores
Amostra 1	200	601
Amostra 2	200	582
Amostra 3	200	599

A quantidade de valores integrados em cada amostra (coluna "# valores") varia de acordo com o número de valores vazios e as repetições presentes em seus *tweets*. O tempo de processamento do componente é avaliado da mesma forma que no componente de mapeamento, por intermédio da execução do mesmo sobre diversos conjuntos de *tweets*. A diferença é que o tempo médio para este componente é medido em minutos, devido aos tempos de execução obtidos. O *Data Warehouse* utilizado é o mesmo do componente de mapeamento.

5.1.2 Métricas para a Avaliação

As métricas utilizadas no decorrer dos experimentos e processo de validação foram aquelas frequentemente utilizadas em procedimentos de recuperação de informação: *recall*, *precision* e *F-measure*. Tais medidas são amplamente utilizadas em aplicações cujas classes são quantitativamente desbalanceadas e/ou a correta predição de uma delas é mais significativa do que a das outras. As Equações em 5.1 e 5.2 representam as fórmulas de *precision* e *recall*, respectivamente.

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

Os termos `TP`, `TN`, `FP` e `FN` são utilizados no cálculo de diversas medidas de validação, sendo aqui obtidos através da matriz de confusão, a qual é largamente utilizada para análise de acertos e erros de algoritmos de classificação. A Tabela 8 representa a estrutura da matriz de confusão e cada um dos termos obtidos a partir dela são definidos como segue:

Tabela 8: Matriz de confusão

	Pred=pos	Pred=neg
Classe=pos	TP	FN
Classe=neg	FP	TN

- ***True positive* (TP):** número de atributos dos *tweets* que foram mapeados para o atributo correto do *DW*, quando se tratando do mapeamento; número de valores que foram integrados com a tupla correta, para o componente de integração.
- ***True negative* (TN):** número de atributos dos *tweets* que não tinham nenhum mapeamento no *DW* e não foram de fato mapeados em nenhum lugar, para o mapeamento; número de valores que não tinham tuplas equivalentes no *DW* e não foram integrados, quando se tratando do componente de integração.
- ***False positive* (FP):** número de atributos transitórios que foram mapeados para algum atributo do *DW* e não deveriam; número de valores dos *tweets* que não tinham tupla equivalente no *DW* mas foram integrados a alguma.
- ***False negative* (FN):** número de atributos transitórios que não foram mapeados para nenhum atributo estacionário mas deveriam ter sido; número de valores transitórios que não foi encontrada tupla equivalente no *DW* mas a mesma existia.

Contextualmente, a métrica de *precision* é definida, para uma determinada classe `C`, como a porcentagem de documentos dentre todos aqueles que foram detectados como pertencentes a esta classe que realmente o são. A métrica de *recall*, por sua vez, representa a porcentagem de documentos, dentre todos aqueles que deveriam ter sido classificados como pertencentes a classe `C`, que realmente o foram. Dessa forma, ambas as medidas assumem valores no intervalo [0,1] e possuem uma relação inversamente proporcional entre si (HAUSSER, 2014). Um componente de mapeamento, por exemplo, com alto valor de *precision* irá retornar poucos mapeamentos dentro os existentes, mas em sua maioria corretos. Enquanto que o mesmo componente com alto valor de *recall* irá retornar um número maior de mapeamentos corretos mas também aumentará o número de mapeamentos incorretos.

Devido a isso, a *F-measure* é largamente utilizada nesses casos por ser uma combinação dessas duas primeiras em uma única métrica, calculada através da média harmônica entre as duas. A fórmula genérica da *F-medida* é dada pela Equação em 5.3, onde β é o parâmetro que determina o peso de cada uma das métricas e é definido de acordo com qual delas deve ser priorizada.

$$F_\beta = \frac{(1+\beta^2) * Precision * Recall}{\beta^2 * Precision + Recall} \tag{5.3}$$

O valor mais comumente utilizado para β é 1, resultando na $F1$-*measure* (Equação 5.4), que atribui pesos iguais à ambas as métricas, balanceando a fórmula. Habitualmente, para os casos em que a métrica de *precision* tem prioridade, é utilizada a $F_{0.5}$, fazendo com que tal medida tenha peso 2 em relação a medida de *recall*. Já para os casos em que a medida de *recall* é aquela de maior importância, a $F2$ é utilizada, ocasionando o efeito contrário (HAN; KAMBER; PEI, 2012).

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5.4}$$

Na presente avaliação experimental, a *F-measure* é utilizada para avaliação do desempenho semântico de ambos os componentes do operador Drill-Conformed. A decisão do valor de β a ser utilizado foi feita a partir da resposta obtida para o seguinte questionamento: "O que é mais importante na integração da informação quando se tratando de tomada de decisão?"

a) obter um maior número de mapeamentos corretos, ao custo de obter juntamente um número considerável de mapeamentos equivocados (maior $recall = F2$);

b) minimizar o número de mapeamentos errados, ao custo de deixar de fora alguns mapeamentos corretos (maior $precision = F_{0.5}$);

c) um equilíbrio entre as opções **a)** e **b)** ($F1$).

Quando analisado o cenário de decisões de negócios, onde as técnicas de suporte à tomada de decisão se aplicam e onde as informações obtidas por esse processo serão utilizadas, dois pontos são considerados importantes: o volume e a exatidão das informações sendo utilizadas. Um volume abrangente em relação ao contexto é considerado satisfatório pois viabiliza uma visão mais ampla do cenário sendo observado. A exatidão, por sua vez, evita que decisões sejam tomadas com base em informações não verídicas ou distorcidas, o que, quando se tratando de negócios, pode gerar um prejuízo considerável para a instituição.

Idealmente, um cenário que consegue detectar todas as informações semânticas que podem ser utilizadas sem detectar dados equivocados juntos seria o ideal. Porém, tal condição pode ser difícil de se alcançar, principalmente quando se lida com extração de dados Web, devido às suas características de dados não estruturados. Dessa forma, foi preciso analisar qual dos dois aspectos traz mais impacto negativo à tomada de decisão: o uso de informações semânticas equivocadas ou o não uso de algumas informações semânticas corretas. Ou se ambas têm o mesmo impacto.

O uso de informações não corretas foi considerado aquele de maior impacto negativo num processo de tomada à decisão, uma vez que ele pode fornecer uma visão do cenário sendo abordado bem mais distorcida do que o não uso de algumas informações. Dessa forma, a precisão do algoritmo foi priorizada e a opção **b)** escolhida. Consequentemente, a métrica $F_{0.5}$ foi utilizada, que, como explicado, desbalanceia a F_β a favor da medida de *precision*. A Equação 5.5 apresenta a fórmula utilizada.

$$F_{0.5} = \frac{1,25 * Precision * Recall}{0,25 * Precision + Recall} \tag{5.5}$$

Para os experimentos e validações que visam analisar o tempo de processamento dos processos executados, as métricas utilizadas para comparação foram as unidades de segundos e minutos.

5.1.3 Configuração da máquina

Uma vez que a arquitetura proposta visa ser uma arquitetura de tomada de decisão em tempo próximo ao real e, consequentemente, validações em cima do tempo de processamento dos componentes foram realizadas, é importante apresentar a configuração da máquina sobre a qual tais experimentos foram executados:

- Processador Intel Core i7 3ª geração - CPU 2.90GHz - 2 núcleos (4 *threads*)
- 8GB de memória ram DDR3
- 1TB de armazenamento interno (disco rígido)

Vale ressaltar também que apenas uma máquina foi utilizada.

5.2 Experimentos e resultados

Nessa seção são apresentados os resultados obtidos para cada um dos experimentos realizados.

5.2.1 DeTEC

A Tabela 9 apresenta a porcentagem de acertos obtidos para cada um dos possíveis rótulos (Tabela 4) em cada uma das categorias apresentadas na Tabela 5. A escolha de utilização da métrica de precisão apenas se deve ao fato de que, não é possível analisar a quantidade de *tweets* relacionados à *streaming* que deveriam ter sido retornados e não o foram, uma vez que se tem acesso apenas aos *tweets* que a consulta retornou como sendo relacionados à mesma. O cálculo da medida de precisão desconsiderou os *tweets* rotulados como "não identificado", uma vez que não é possível definir se o mesmo é um *true positive* ou *false positive*.

Tabela 9: Resultados DeTEC - Detecção e extração

Categoria	**# *tweets***	**Relacionado**	**Não relacionado**	**Não identificado**	**Precision**
1	200	26,5%	73%	0,5%	0,265
2	107	72%	21%	7%	0,773
3	317	99,8%	0%	1,2%	1,000
4	226	98,2%	0,9%	0,9%	0,991
5	155	95,8%	1,3%	2,9%	0,986
6	41	87,1%	1,4%	1,5%	0,983
7	100	57%	35%	8%	0,619
Todos	1.146	78,7%	17,9%	3,4%	0,814

Como pode ser observado, o ponto crítico do algoritmo de extração se encontra em *streamings* cujo título se encontram na Categoria 1, que são aqueles compostos de uma única palavra utilizada frequentemente e de muitas maneiras no cenário de *streamings*, obtendo uma taxa de precisão de apenas 26,5%. Em seguida, encontra-se a Categoria 7, composta de *streamings* com nomes grandes que representam expressões do dia-a-dia e podem ser utilizados em diversos contextos, com precisão no valor de 61,9%. Por outro lado, nota-se que as demais categorias alcançaram um bom desempenho em relação a quantidade de dados obtidos corretamente, a maioria com índice acima de 90%.

Para o tempo de processamento, dois processos foram verificados: i) o tempo requerido para detecção e extração dos *tweets*; ii) o tempo de processamento da *API* de análise de sentimento utilizada no desenvolvimento deste trabalho. A etapa de carga dos dados não foi avaliada porque é realizada de maneira instantânea. A Tabela 10 apresenta os valores obtidos para cada processo.

Tabela 10: Resultados DeTEC - Tempos de processamento

Processo	**# *tweets***	**Tempo médio**
detecção e extração	11.236	7 minutos
análise de sentimento	11.236	31 segundos

Para verificar o tempo de processamento foi executada uma consulta sobre o protótipo criado. Tal consulta retornou 11.236 *tweets* – que é a média de *tweets* retornados para *streamings* que têm sido bastante comentadas num período recente – e demorou aproximadamente 7 minutos para realizar a detecção e extração de todos esses *tweets*. Verificou-se também, que a *API* sentiment140, utilizada na etapa de transformação para execução do processo de análise de sentimento dos *tweets* obtidos, analisa essa quantidade de itens em um tempo médio de 31 segundos, o que é considerado um resultado positivo quando avalia-se sua execução no cenário de tempo real.

Dessa forma, o processo de DeTEC implementado neste trabalho leva em média 8 minutos para ser executado em um conjunto de aproximadamente 11 mil *tweets*, levando em conta a carga dos dados. Considerando o cenário de disponibilização de *streamings*, sobre o qual esse projeto foi aplicado, e as definições de *Real Time ETL* apresentadas por Vaisman e Zimányi (2012) – que diz que análise de informações de uso de sistemas Web e cenários de recomendação e colaboratividade tem uma janela de atualização do *DW* no intervalo de horas, sem causar grandes prejuízos à instituição – tal resultado foi considerado satisfatório.

Além disso, a partir das buscas realizadas em diversos períodos e com *streamings* das diversas categorias apresentadas, constatou-se que a quantidade de *tweets* retornados não tende a exceder essa média de itens. Assim sendo, o tempo de processamento do processo de DeTEC não está propenso a ser muito maior que isso. Adicionalmente, como apontado por Etcheverry e Vaisman (2012), dados Web concentrados em um determinado assunto e momento, no geral, não tendem a atingir volumes muito excedentes aos utilizados nesta avaliação.

Vale ressaltar, porém, que esse tempo pode variar dependendo do estudo de caso ao qual for aplicado e das transformações a serem realizadas sobre os dados, além da latência da rede. Assim como o limiar que considera o desempenho temporal obtido como satisfatório irá variar de acordo com o ambiente no qual o sistema de suporte à decisão é implementado (VAISMAN; ZIMÁNYI, 2012). Além disso, o processo de DeTEC executado omite alguns passos de transformação e enriquecimento, que podem variar de acordo com o contexto. Dessa forma, esse tempo pode vir a ser um pouco maior, ou menor, dependendo do processo realizado.

5.2.2 Componente de mapeamento

A Tabela 11 apresenta os resultados da busca em *grid* obtidos para cada um dos valores atribuídos ao parâmetro `perc`.

Tabela 11: Busca em *grid* - Porcentagem mínima de votos (`perc`)

	2.000 *tweets*			**10.000 *tweets***		
Votos %	**precision**	**recall**	$F_{0.5}$	**precision**	**recall**	$F_{0.5}$
10	0,875	0,875	0,875	0,888	1,000	0,909
20	0,875	0,875	0,875	0,875	0,875	0,875
30	0,875	0,875	0,875	0,875	0,875	0,875
40	0,875	0,875	0,875	0,875	0,875	0,875
50	0,875	0,875	0,875	**1,000**	**0,750**	**0,937**
60	**1,000**	**0,750**	**0,937**	**1,000**	**0,750**	**0,937**
70	1,000	0,625	0,892	1,000	0,625	0,892
80	1,000	0,625	0,892	1,000	0,500	0,833

Analisando os resultados apresentados, podemos observar que o melhor parâmetro a ser utilizado para `perc` é o de 60%, pois foi aquele que maximizou a $F_{0.5}$ tanto para o conjunto de 2.000 *tweets* quanto para aquele de 10.000. A queda de desempenho para os valores acima de 60% se deve ao fato de que, quando maior a porcentagem de votos necessários, menor é o número de mapeamentos considerados válidos, diminuindo a medida de *recall*. É possível observar também que os valores possuem resultados equilibrados entre os dois conjuntos de *tweets*, o que é um *feedback* considerado positivo, uma vez que a quantidade de *tweets* pode variar de acordo com as buscas realizadas e o componente deve se manter estável diante deste fator.

Após estabelecer a porcentagem mínima de votos a ser considerada na execução do componente de mapeamento, o mesmo foi executado sobre os *datasets* apresentados na Tabela 6, a fim de validar o seu desempenho semântico num conjunto de dados mais abrangente e não somente sobre a estrutura dos *tweets*. O critério de escolha dos *datasets* foi a existência de atributos que pudessem ser mapeados no modelo de *DW* criado para o estudo de caso sobre o qual este projeto foi desenvolvido. Os resultados obtidos para cada um deles, e para o conjunto de *tweets*, são expostos na Tabela 12.

Como pode-se constatar, o valor de *precision* foi maior para todos os *datasets*. Isto se deve ao fato de que o valor de parâmetro escolhido anteriormente priorizava a medida de *precision* em contrapartida à de *recall*. Pode-se notar também que, para o *dataset* denominado "Bancos", o desempenho do componente caiu quando comparado com os demais *datasets*. Tal resultado é consequência da estrutura dos dados contidos nesse grupo, que consiste em poucos documentos JSONs contendo muitos atributos e onde a grande maioria deles são campos de texto livre. A baixa cardinalidade do conjunto faz com

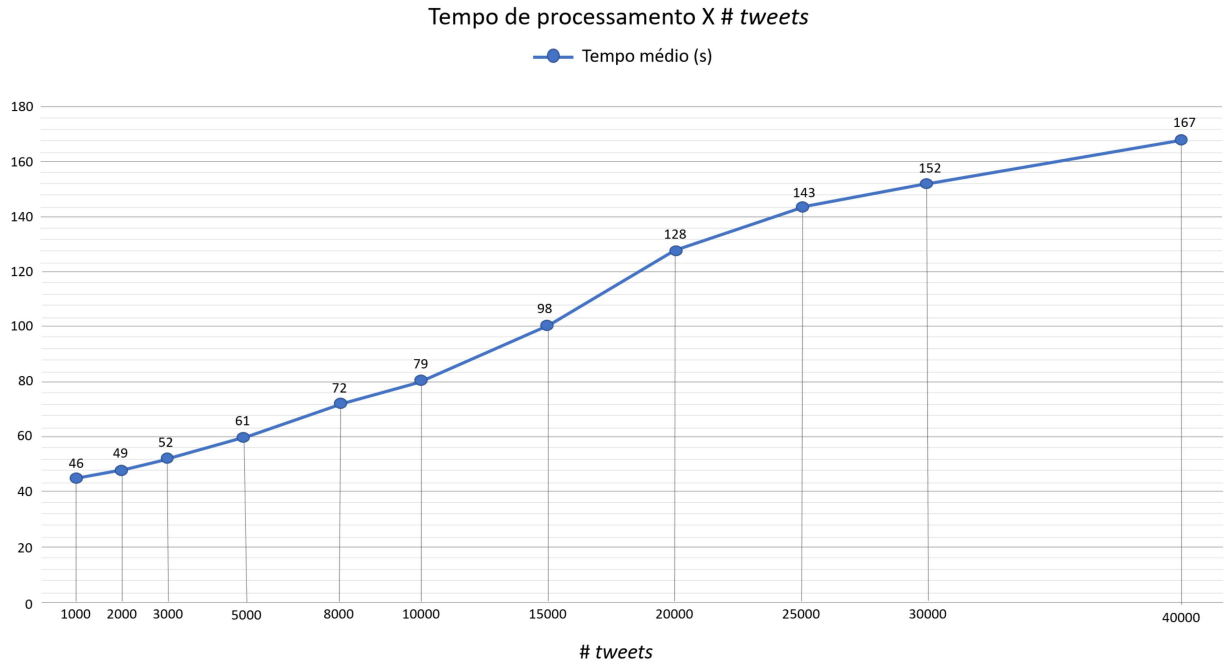
Tabela 12: Resultados mapeamento - semântica

Dataset	**precision**	**recall**	$F_{0.5}$
Tweets	1,000	0,750	0,937
Aeroportos	1,000	0,800	0,952
Zip Codes	1.000	0.666	0.908
Bancos	0,727	0,625	0,705

que o número necessário de votos seja menor e, combinado com a grande quantidade de campos de texto aberto, a distinção da coocorrência dos conjuntos se torna menos ampla. Ainda assim, o desempenho de 70% obtido, apesar de abaixo da média dos demais, não é considerado um desempenho insatisfatório (GIACOMETTI; MARCEL; NEGRE, 2008).

Por fim, a Figura 17 apresenta o gráfico que expõe o tempo de processamento, em segundos, obtido pelo componente para conjuntos de *tweets* com cardinalidades diversas.

Figura 17: Resultados mapeamento - tempo de processamento



Fonte: Produzido pela autora

Apesar de o tempo considerado próximo do real variar de acordo com as necessidades do negócio sobre o qual a tomada de decisão é realizada, o desempenho computacional apresentado pelo componente de mapeamento foi considerado extremamente satisfatório. Analisando-se os resultados obtidos é possível notar que os tempos se mantém conside-

ravelmente baixos mesmo com o aumento do volume de dados. Essa é considerada uma característica positiva, quando considerado o fato de que o volume de um *DW* cresce continuamente.

5.2.3 Componente de integração

As Tabelas 13, 14 e 15 exibem as matrizes de confusão geradas por meio da classificação dos valores selecionados para as Amostras 1, 2 e 3, respectivamente. Os resultados obtidos por intermédio das mesmas são apresentados na Tabela 16. A utilização da $F_{0.5}$ é justificada pelos mesmos motivos expostos na Seção 5.2.2.

Tabela 13: Matriz de confusão - Amostra 1

	Pred=pos	Pred=neg
Classe=pos	387	10
Classe=neg	19	185

Tabela 14: Matriz de confusão - Amostra 2

	Pred=pos	Pred=neg
Classe=pos	385	27
Classe=neg	14	156

Tabela 15: Matriz de confusão - Amostra 3

	Pred=pos	Pred=neg
Classe=pos	383	12
Classe=neg	20	184

Tabela 16: Resultados integração - semântica

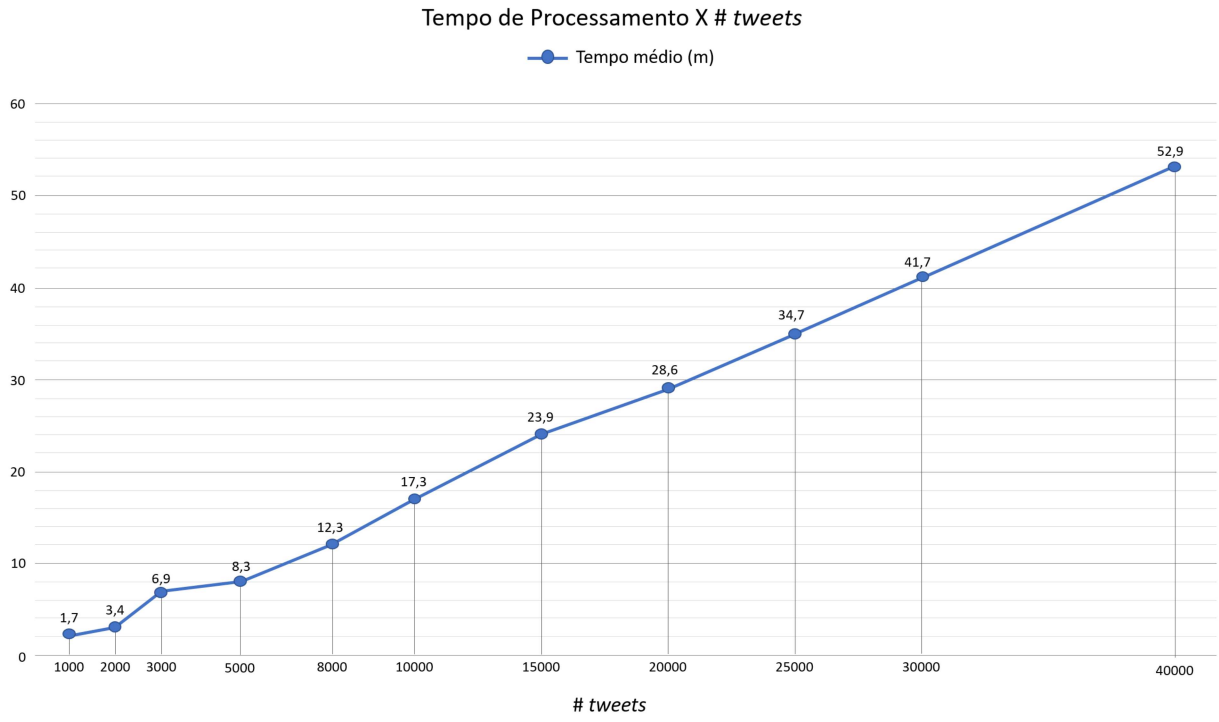
Amostra	**precision**	**recall**	$F_{0.5}$
Amostra 1	0,953	0,974	0,957
Amostra 2	0,964	0,934	0,957
Amostra 3	0,950	0,969	0,953

Com base nos resultados obtidos, apresentados da Tabela 16, pode-se verificar que o operador mantém um desempenho estável para todas as amostras sobre o qual foi aplicado. Também, é possível constatar que o mesmo teve um alto desempenho na

integração de todas as amostras, sempre acima de 95% de acerto. Esse resultado foi considerado satisfatório ao propósito do mesmo.

Por último, o tempo de processamento do componente foi avaliado para grupos de *tweets* com cardinalidades distintas, da mesma forma que foi realizado para o componente de mapeamento. A Figura 18 apresenta o gráfico que mostra os tempos obtidos, em minutos, para cada grupo, considerando que, para cada *tweet*, 4 de seus atributos foram integrados aos dados estacionários. Além disso, a integração foi feita sobre um ambiente de sistema de gerenciamento de banco de dados relacional, utilizando índices para os atributos do *Data Warehouse*.

Figura 18: Resultados integração - tempo de processamento



Fonte: Produzido pela autora

Para o estudo de caso deste projeto, os tempos obtidos pelo componente são considerados aceitáveis, uma vez que, como já mencionado mais acima, a tomada de decisão para este cenário permite uma janela de atualização considerável no que é tido como tempo próximo do real. Porém, quando analisado do ponto de vista que o operador proposto tem o intuito de ser aplicado aos mais diversos casos, o desempenho obtido pode vir a não ser adequado, dependendo do cenário. Os valores alcançados são consequência do fato de que, para realizar a integração, é necessário percorrer os atributos e itens a serem integrados um a um e, de alguma forma, compará-los ao conjunto de valores do *DW*, a fim

de encontrar a tupla correspondente. Diferentemente do processo de mapeamento, onde os dados podem ser comparados em conjuntos, a integração necessita realizar a comparação item a item, uma vez que o intuito é encontrar a melhor combinação. Realizar esse processo de maneira rápida, considerando as características de um *DW*, cujo número de atributos é, geralmente, alto e o volume de dados tende a ser cada vez maior, é um desafio a ser encarado. A utilização de técnicas de paralelismo e/ou processamento distribuído podem ser bons candidatos a serem estudados para melhorar este desempenho.

5.3 Sumário

Esse capítulo apresentou os três experimentos realizados para avaliação da proposta apresentada. A partir das experimentações, pôde-se concluir que todos os processos avaliados obtiveram bons resultados quando se tratando do aspecto semântico. Com relação ao tempo de processamento, ponto importante a ser considerado devido à característica da arquitetura proposta, a maioria dos componentes atingiram valores satisfatórios para o volume de dados utilizado.

O processo de DeTEC obteve bons resultados, considerando o estudo de caso aplicado, para ambos os aspectos avaliados. Todavia, é importante ressaltar que algumas etapas do processo de transformação foram omitidas, por não serem o foco deste projeto experimental, e que o processo de DeTEC pode variar de acordo com o domínio de dados sobre o qual o sistema de suporte à decisão é aplicado.

Para o componente de mapeamento, foi verificado que o valor de 60% para a porcentagem mínima de votos é o que obtém melhores resultados, levando em consideração a precisão do processo. Porém, esse valor pode ser mudado dependendo das necessidades e características dos dados sendo integrados e/ou preferências do usuário. O tempo de processamento desse componente também foi considerado extremamente satisfatório, levando menos de 3 minutos para conjuntos de 40 mil dados e um *Data Warehouse* de 81MB.

O componente de integração do operador *OLAP* proposto, apesar de ter alcançado um alto desempenho no seu aspecto semântico, com valores sempre acima de 90%, possui um desempenho computacional que pode não ser satisfatório dependendo do cenário ao qual o mesmo será aplicado. Dessa forma, é interessante pensar em medidas que possam diminuir essa latência, dentre as quais podem ser boas opções o uso de processamento distribuído ou técnicas de computação paralela.

Por fim, vale ressaltar que, a proposta, de maneira geral, não foi comparada com propostas similares pois, até o momento de finalização deste projeto, a autora não tem conhecimento da existência de nenhum trabalho similar implementado e contendo resultados experimentais.

6 Conclusão

Respondendo à primeira questão levantada no Capítulo 1, que diz respeito a capacidade de obtenção e integração dos dados de maneira automática, a fim de permitir consultas que envolvam dados estacionários e transitórios, é possível inferir que esse objetivo foi devidamente atingido, obtendo bons resultados. No que diz respeito ao segundo questionamento, que se refere a capacidade deste processo ser realizado em tempo próximo ao real, quando unificando o desempenho computacional de todos os elementos em um único componente, o objetivo foi atingido parcialmente, dependendo do cenário sobre o qual a arquitetura é instanciada. Para que esse objetivo seja completamente auferido para qualquer contexto, é necessário que sejam ultrapassados alguns desafios ainda.

Com base nos trabalhos estudados, foi possível notar que ainda não existem muitas propostas não teóricas em relação ao que foi apresentado neste trabalho, principalmente quando se tratando de operadores *OLAP* capazes de lidar com dados semi ou não estruturados de maneira automática e utilização de mídias sociais. O trabalho exposto trabalhou sobre esses pontos, por meio da utilização do Twitter como fonte de dados e desenvolvimento do Drill-conformed.

A arquitetura proposta, por se tratar de uma arquitetura genérica, pode ser utilizada em diversos cenários e como base para futuros estudos e pesquisas relacionados a essa nova geração de *BI*, que vem crescendo bastante nos últimos anos. Para o operador, a partir do estudo de caso e protótipo utilizados, foi possível verificar que bons resultados podem ser obtidos, mas algumas melhorias também podem ser realizadas.

De maneira geral, os objetivos inicialmente definidos foram atingidos durante o desenvolvimento deste projeto. Existem melhorias já citadas a serem realizadas, contudo, com relação ao propósito para o qual este trabalho foi elaborado, e as características às quais foi aplicado, os resultados são tidos como satisfatórios.

Dentre as contribuições oferecidas por este trabalho, destacam-se:

1. Proposta de uma arquitetura condizente com a *BI* 2.0.
2. Criação de um operador *OLAP* capaz de integrar dados semi ou não estruturados com dados estruturados de maneira automática.
3. Utilização de dados da Web como fonte de dados para a tomada de decisão.
4. Utilização de mídias sociais e opinião de usuário na tomada de decisão.
5. Geração de informações semânticas cabíveis de disponibilização na Web semântica.

6. Provisão de autonomia ao usuário.
7. Disponibilização do *dataset* de *tweets* extraídos para a experimentação[1].

6.1 Publicações

Como parte deste trabalho, os seguintes artigos foram desenvolvidos:

- O artigo **SelfBI: Tomada de decisão sob demanda do usuário utilizando dados da Web** foi publicado nos Anais do 31º Simpósio Brasileiro de Banco de Dados (SBBD), realizado em Salvador - BA, em Outubro/2016.
- O artigo **Drill-Conformed: automatic and autonomous decision-making process** foi submetido ao periódico de *Decision Support Systems* da Elsevier, ISSN: 0167-9236, Fator de Impacto: 3,222.

6.2 Trabalhos Futuros

Os trabalhos a serem realizados futuramente incluem:

1. Desenvolver o componente de colaboratividade, a fim de que as informações extraídas pelo componente de mapeamento possam ser úteis à demais usuários na rede.
2. Elaborar uma estratégia para melhorar o tempo de processamento do componente de integração, por intermédio de processamento paralelo ou distribuído.
3. Realizar experimentos acerca da generalidade do operador, aplicando ele a diversos cenários e, se possível, utilizando usuários finais para validação;
4. Aperfeiçoar o operador para ser capaz de trabalhar com os demais tipos de dados não estruturados.
5. Permitir a seleção de fontes de dados transitórios de maneira dinâmica, a partir da consulta do usuário e uma biblioteca de fontes disponíveis juntamente com informações acerca de seus dados.
6. Implementar um mecanismo de recomendação de fontes de dados por parte dos usuários, de acordo com o assunto a ser buscado.

[1] Disponível em: <https://github.com/CaahSilva/streamings-dataset>

Referências

ABELLÓ, A. et al. Fusion cubes: Towards Self-Service Business Intelligence. *International Journal of Data Warehousing and Mining*, v. 9, p. 66–88, 2013. Citado 4 vezes nas páginas 30, 41, 44 e 60.

ABELLÓ, A. et al. Using Semantic Web Technologies for Exploratory OLAP: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, v. 27, p. 571–588, 2015. Citado 2 vezes nas páginas 23 e 40.

BENKER, T. A Hybrid OLAP & OLTP Architecture Using Non-Relational Data Components. In: *Enterprise Modelling and Information Systems Architectures: Proceedings of the 5th International Workshop on Enterprise Modelling and Information Systems Architectures, EMISA*. St. Gallen, Switzerland: GI, 2013. (LNI, v. 222), p. 41–57. Disponível em: <http://subs.emis.de/LNI/Proceedings/Proceedings222/41.pdf>. Acesso em: 21 jan. 2017. Citado na página 45.

BERRO, A.; MEGDICHE, I.; TESTE, O. A Content-Driven ETL Processes for Open Data. In: *New Trends in Database and Information Systems II*. Ohrid, Macedonia: Springer International Publishing, 2015. (Advances in Intelligent Systems and Computing, v. 312), p. 29–40. Disponível em: <https://www.academia.edu/17019811/A_Content-Driven_ETL_Processes_for_Open_Data>. Acesso em: 26 jan. 2017. Citado na página 48.

BOODIDHI, S. *Using smoothing techniques to improve the performance of Hidden Markov's Model*. Dissertação (Mestrado) — University of Nevada, Las Vegas, 2011. Citado na página 65.

CASTELLANOS, M. et al. A platform for situational awareness in operational BI. *Decision Support Systems*, v. 52, p. 869–883, 2012. Citado na página 42.

CHAUDHURI, S.; DAYAL, U. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, New York, NY, USA, v. 26, p. 65–74, 1997. Citado na página 29.

CHEN, H.; CHIANG, R. H. L.; STOREY, V. C. Business Intelligence and Analytics: From Big Data to Big Impact. *Management Information Systems Quarterly*, v. 36, p. 1165–1188, 2012. Citado 2 vezes nas páginas 24 e 39.

DEAN, J.; GHEMAWAT, S. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, v. 51, p. 107–113, 2008. Citado na página 47.

ETCHEVERRY, L.; VAISMAN, A. A. Enhancing OLAP Analysis with Web Cubes. *Lecture Notes in Computer Science*, v. 7295, p. 469–483, 2012. Citado 4 vezes nas páginas 24, 40, 41 e 103.

FOWLER, A. *NoSQL For Dummies*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2015. Citado na página 57.

GIACOMETTI, A.; MARCEL, P.; NEGRE, E. A Framework for Recommending OLAP Queries. In: *Proceedings of the ACM 11th international workshop on Data warehousing and OLAP, (DOLAP '08)*. Napa Valley, California, USA: ACM New York, NY, USA,

2008. p. 73–80. Disponível em: <http://dl.acm.org/citation.cfm?id=1458446>. Acesso em: 10 jul. 2017. Citado na página 105.

GONZÁLEZ, S. M.; BERBEL, T. Considering unstructured data for OLAP: a feasibility study using a systematic review. *Revista de Sistemas de Informação da FSMA*, v. 14, p. 26–35, 2014. Citado 2 vezes nas páginas 24 e 40.

GUPTA, G. K. *Introduction to Data Mining with Case Studies*. 3. ed. New Delhi: PHI Learning, 2014. Citado na página 39.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3. ed. United States of America: Morgan Kaufmann, 2012. Citado 8 vezes nas páginas 29, 31, 32, 33, 34, 35, 38 e 100.

HAUSSER, R. *Foundations of Computational Linguistics: Human-Computer Communication in Natural Language*. Erlangen, Germany: Springer-Verlag Berlin Heidelberg, 2014. Citado na página 100.

HOLUPIREK, A.; GRÜN, C.; SCHOOL, M. H. BaseX & DeepFS Joint Storage for Filesystem and Database. In: *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (EDBT)*. Saint Petersburg, Russia: ACM New York, NY, USA, 2009. p. 1108–1111. Disponível em: <https://goo.gl/kii7rf>. Acesso em: 22 jan. 2017. Citado na página 47.

IGLESIAS, J. A. et al. Web news mining in an evolving framework. *Information Fusion*, v. 28, p. 90–98, 2015. Citado na página 23.

INMON, W. H. *Building the Data Warehouse*. 3. ed. New York, NY, USA: John Wiley & Sons, Inc., 2002. Citado na página 29.

JENSEN, C. S.; PEDERSEN, T. B.; THOMSEN, C. *Multidimensional Databases and Data Warehousing*. Denmark: Morgan & Claypool Publishers series Synthesis Lectures on Data Management, 2010. (Synthesis Lectures on Data Management). Citado 3 vezes nas páginas 28, 30 e 33.

JÖRG, T.; DESSLOCH, S. Near Real-Time Data Warehousing Using State-of-the-Art ETL Tools. *Lecture Notes in Business Information Processing*, v. 41, p. 100–117, 2010. Citado na página 30.

KIMBALL, R.; ROSS, M. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. 3. ed. Indianopolis, Indiana: John Wiley & Sons, Inc., 2013. Citado 6 vezes nas páginas 28, 31, 32, 33, 34 e 35.

KLYNE, G.; CARROLL, J. J.; MCBRIDE, B. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. 2004. Disponível em: <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>. Citado na página 41.

MANSMANN, S. et al. Discovering OLAP dimensions in semi-structured data. *Information Systems*, v. 44, p. 120–133, 2014. Citado 3 vezes nas páginas 23, 47 e 59.

PASSLICK, J.; LEBEK, B.; BREITNER, M. H. A Self-Service Supporting Business Intelligence and Big Data Analytics Architecture. In: *Proceedings of the 13th International Conference on Wirtschaftsinformatik, (WI 2017)*. St. Gallen, Switzerland: [s.n.],

2017. (Towards Thought Leadership in Digital Transformation). Disponível em: <http://aisel.aisnet.org/wi2017/track12/paper/5/>. Acesso em: 11 mar. 2017. Citado na página 50.

PRUD'HOMMEAUX, E.; SEABORNE, A. *SPARQL Query Language for RDF*. 2008. Disponível em: <http://www.w3.org/TR/rdf-sparql-query/>. Citado na página 42.

SADALAGE, P. J.; FOWLER, M. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. 1. ed. Crawfordsville, Indiana: Pearson Education, Inc, 2013. Citado na página 82.

THOMSEN, C.; PEDERSEN, T. B.; LEHNER, W. RiTE: Providing On-Demand Data for Right-Time Data Warehousing. In: *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE)*. Cancun, Mexico: IEEE, 2008. p. 456–465. Disponível em: <http://ieeexplore.ieee.org/document/4497454/>. Acesso em: 02 jul. 2017. Citado 2 vezes nas páginas 24 e 40.

TRUJILLO, J.; MATÉ, A. Business Intelligence 2.0: A General Overview. *Lecture Notes in Business Information Processing*, v. 96, p. 98–116, 2012. Citado 2 vezes nas páginas 24 e 39.

VAISMAN, A.; ZIMÁNYI, E. Data Warehouses: Next Challenges. *Lecture Notes in Business Information Processing*, v. 96, p. 1–26, 2012. Citado 3 vezes nas páginas 23, 39 e 103.