**UNIVERSIDADE FEDERAL DE SÃO CARLOS**
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# INTUITIVE: MODELO CONCEITUAL PARA *WORKFLOWS* DE ETL

**ANA CÉLIA RIBEIRO BIZIGATO PORTES**

**ORIENTADOR: PROF. DR. RICARDO RODRIGUES CIFERRI**

São Carlos - SP

Setembro/2020

**UNIVERSIDADE FEDERAL DE SÃO CARLOS**
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# INTUITIVE: MODELO CONCEITUAL PARA *WORKFLOWS* DE ETL

**ANA CÉLIA RIBEIRO BIZIGATO PORTES**

Monografia apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Engenharia de Software, Banco de Dados e Interação Humano Computador

Orientador: Dr. Ricardo Rodrigues Ciferri

São Carlos – SP

Setembro/2020

**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

## Folha de Aprovação

Defesa de Dissertação de Mestrado da candidata Ana Célia Ribeiro Bizigato Portes, realizada em 09/09/2020.

### Comissão Julgadora:

Prof. Dr. Ricardo Rodrigues Ciferri (UFSCar)

Prof. Dr. Renato Bueno (UFSCar)

Profa. Dra. Valéria Cesario Times (UFPE)

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

*Para os meus filhos, Júlio e Anita, para lembra-los de que é possível aprender em qualquer tempo.*

# AGRADECIMENTO

Agradeço muito ao Professor Ricardo Ciferri, meu orientador e meu amigo, pela dedicação, pela paciência, pelo comprometimento e pela generosidade em compartilhar seu conhecimento. Agradeço também pela amizade de longa data.

Agradeço ao DC–UFSCar que foi minha casa por duas temporadas extraordinárias: minha graduação e meu mestrado. Tenho muito orgulho de ser DC-UFSCar!

Agradeço também à Professora Cristina Ciferri, pelas valiosas contribuições.

Agradeço aos meus pais, Joana e Fiore, por sempre acreditarem que a educação é o grande o poder transformador da sociedade. E, por fim, agradeço ao Milton, meu marido e companheiro de uma vida inteira, pelo apoio e o incentivo sem os quais esse trabalho não teria sido possível.

*"Um dia é preciso deixar de sonhar, tirar os planos da gaveta e, de algum modo, começar."*

*Amyr Klink*

# RESUMO

O domínio da informação é visto como um diferencial competitivo nas mais variadas áreas de negócio, tais como na saúde, agronegócio, telecomunicações, logística e em órgãos governamentais. A informação correta e atualizada é um valioso subsídio para decisões estratégicas nas corporações. Soma-se a isso o fato de que, atualmente, imensos volumes de dados são gerados em alta velocidade e em diversos formatos. Nesse contexto, pesquisas têm sido realizadas com o objetivo de propor novos modelos, arquiteturas, processos e algoritmos que possam contribuir para a transformação dos dados em informações úteis para a tomada de decisão estratégica. Nesse cenário, um ambiente de *data warehousing* exerce um papel fundamental. Esse ambiente contém o *data warehouse* (DW), que é o grande repositório que armazena dados extraídos de diversas fontes e que foram devidamente tratados e acurados. Os dados contidos no DW são usados para responder a consultas OLAP (*Online Analytical Processing*). Em um ambiente de *data warehousing*, o processo de ETL é usado para a extração dos dados brutos das diversas fontes de dados, seguido das etapas de transformação, limpeza e integração desses dados, para no final prover o armazenamento dos dados acurados no DW. Além da carga inicial dos dados, o pesquisa de ETL é usado para a constante atualização dos dados no DW. Esta pesquisa de Mestrado investigou as melhores práticas utilizadas na modelagem conceitual de *workflows* de ETL e, como resultado, propõe um novo modelo, denominado "Intuitive", que adiciona simplicidade, agilidade, clareza e consistência à etapa de modelagem, podendo contribuir para melhorar a construção e a manutenção de *workflows* de ETL. Para a validação do modelo Intuitive forma realizadas atividades de análise teórica e, também, experimentos práticos com a participação de usuários. Tais atividades permitiram avaliar o modelo Intuitive, cujos elementos se mostraram suficientes para representar com clareza diversos cenários típicos de ETL demonstrando vantagens quando comparado ao principal trabalho relacionado no estado da arte.

**Palavras-chave**: *data warehouse*, ETL, modelagem, modelagem conceitual, *workflow*.

# ABSTRACT

The information domain is seen as a competitive differential in the most varied business areas, such as health, agribusiness, telecommunications, logistics, and government agencies. The correct and updated information is a valuable subsidy for corporative strategic decisions. Additionally, nowadays, huge volumes of data are generated at high speed and in various formats. In this context, research has been made to propose new models, architectures, processes, and algorithms that can contribute to transforming data into useful information for strategic decision making. In this scenario, a data warehousing environment plays a key role. The environment contains the data warehouse (DW), a huge repository with data that serves as a basis for responding to OLAP (Online Analytical Processing) queries. In a data warehousing environment, the ETL process is used to extract raw data from different data sources and to transform, clean, and integrate that data, loading to the DW. The ETL process is used for first data loading and, also for refreshing the data in the DW. This master's research investigated the best practices in conceptual modeling for ETL workflows and, as a result, proposes a new model, called “Intuitive”. The Intuitive Model adds simplicity, agility, clarity, and consistency to the modeling stage and can contribute to the improvement of construction and maintenance of ETL workflows. Theoretical analysis activities and practical experiments were performed with the users’ participation in order to validate the Intuitive Model. Such steps allowed us to evaluate that the elements of the Intuitive Model are sufficient to represent clearly several regular ETL scenarios showing advantages in comparison with the main related work in the state of the art.

**Keywords**: data warehouse, ETL, modeling, conceptual modeling, workflow.

# SUMÁRIO

# Capítulo 1

# INTRODUÇÃO

*Neste capítulo são apresentados a contextualização, a motivação e os objetivos dessa pesquisa de Mestrado. Na motivação são descritos os principais desafios relacionados à construção, ao tratamento e à manipulação de workflows de ETL como parte da construção de um ambiente de data warehousing. Nos objetivos são apresentados os questionamentos que nortearam a realização dessa pesquisa.*

## 1.1 **Contexto**

Nos dias atuais, o domínio da informação pode ser visto como um diferencial competitivo nas mais variadas áreas de negócio. A informação correta e atualizada é um valioso subsídio para a tomada de decisão estratégica nas corporações. Dados sumarizados e integrados são usados pelos gestores para embasar decisões e propostas de ações, tais como lançamento de novos produtos, ajustes de preços e lançamento de promoções, implementação de políticas públicas, distribuição de recursos, priorização de investimentos, analisando os dados históricos e as tendências (CHAUDHURI; DAYAL, 1997).

*Data warehouse* (DW) é o termo usado para descrever o grande repositório de dados integrados que são obtidos a partir de provedores de informação

autônomos, distribuídos e heterogêneos (CIFERRI, 2002; INMON, 2005; INMON; STRAUS; NEUSHLOSS, 2008). Esse repositório pode ser usado para o processamento de consultas OLAP (*Online Analytical Processing*), cujos resultados apoiam o processo de tomada de decisão estratégica em setores tais como na saúde, varejo, agronegócio, telecomunicações, logística, em órgãos governamentais e tantos outros. Assim, é desejável que os dados contidos no DW possam ser facilmente compreendidos pelos usuários que os acessam por meio de consultas OLAP, que sejam úteis, confiáveis, consistentes e que tenham acesso controlado (KIMBALL, 2002).

Um ambiente de *data warehousing* envolve ferramentas, algoritmos e processos que são utilizados para apoiar as etapas de (i) ETL (*Extract, Transform, and Load*), (ii) definição, projeto e criação do DW e (iii) processamento de consultas OLAP. Em um ambiente de *data warehousing*, o processo de ETL é crítico e complexo, porque contempla desde os eventos registrados nas fontes de dados, passando pelo mapeamento dos processos de negócio, até a disponibilização dos dados essenciais para as várias consultas gerenciais que serão posteriormente realizadas sobre o DW.

Essa pesquisa de Mestrado investigou a construção de *workflows* de ETL em ambientes de *data warehousing* com foco na atividade de modelagem conceitual, que enfoca a representação do *workflow* de forma mais abstrata, em mais alto nível e independente de implementação. As soluções usadas para modelagem de ETL foram comparadas usando critérios e requisitos de qualidade e, com base nessa comparação, foi proposto um novo modelo conceitual para *workflows* de ETL, denominado modelo Intuitive.

## 1.2 **Motivação**

Há poucos anos não era possível prever o volume descomunal de dados que são gerados no mundo atual. O crescente número de dispositivos físicos

ligados à internet, o intenso uso de redes sociais, os dados gerados pelos inúmeros sensores e até os satélites contribuem para a produção de um gigantesco volume de dados nos mais diferentes formatos (texto, vídeo, música). De modo geral, tem-se gerado cada vez mais dados, com mais rapidez e nos mais diferentes formatos e formas de armazenamento. Esse contexto tem permitido a descoberta de novos valores e a obtenção de novos conhecimentos que até então estavam ocultos e promove a aplicação de novas tecnologias, algoritmos, processos e ferramentas de análise. Assim, surgem novas oportunidades e desafios para pesquisa e desenvolvimento de soluções que permitam o tratamento, a análise e o armazenamento de dados (CHEN et al., 2014).

Nesse sentido, em um ambiente de *data warehousing* faz-se necessária a utilização de tecnologias e ferramentas que permitam lidar com os dados de forma eficiente, transformando-os em informações úteis para os usuários e para as organizações. Na visão de Kimball (2002), embora seja pouco visível pelos usuários de DW, o processo de ETL é custoso e pode envolver até 70% do esforço da implementação e manutenção do DW. Esse custo está associado (i) ao esforço necessário para a construção e manutenção do sistema, (ii) ao tempo necessário para a execução do processo, (iii) aos recursos computacionais necessários para o processamento dos dados e (iv) e ao custo financeiro (BALA; BOUSSAID; ALIMAZIGH, 2015). Com isso, requisitos de qualidade tais como robustez, capacidade de recuperação de falhas, flexibilidade, disponibilidade, facilidade de manutenção e confiabilidade são bastante importantes para *workflows* de ETL (SIMITSIS et al., 2009).

EL-SAPPAGH et al*.* (2011) afirmam que, para o sucesso de um projeto de DW, deve haver um *workflow* de ETL robusto, bem documentado e que deve ser fácil de ser mantido e, assim, os conceitos associados aos *workflows* de ETL têm sido renovados para lidar eficientemente com o enorme volume de dados e com a grande velocidade de geração desses dados, passando por inovações, modificações e atualizações para obter maior agilidade no processamento, para

facilitar sua implementação e manutenção e, também, para elevar a qualidade dos dados resultantes que são armazenados no DW (BALA; BOUSSAID; ALIMAZIGH, 2015).

Nesse contexto, os investimentos na modelagem de ETL, para representar adequadamente as operações aplicadas aos dados ao longo do *workflow*, é essencial para a construção de um *ambiente de data warehousing*. Além disso, o processo de ETL é também essencial para outras aplicações de suporte a decisão, tais como aplicações de mineração de dados e criação de relatórios: essas aplicações não exigem a modelagem multidimensional dos dados e podem acessar um único arquivo contendo dados tratados, limpos e integrados; portanto, o uso ETL é ainda mais amplo no contexto de soluções de suporte a decisão do que soluções de *data warehousing* que foram o foco dessa pesquisa de Mestrado.

## 1.3 **Objetivos**

Esta pesquisa de Mestrado investigou a construção e a representação de *workflows* de ETL, como parte de um ambiente de *data warehousing*, com foco na etapa de modelagem conceitual. Um objetivo inicial foi realizar o levantamento do estado da arte na solução do problema descrito, com o intuito de entender as suas potencialidades e limitações. Diante do exposto foi definido o seguinte tema de pesquisa: promover a melhoria no processo de construção de *workflows* de ETL em ambientes de *data warehousing* e, dentro do tema estabelecido, buscou-se uma solução para o seguinte problema: como melhorar a atividade de modelagem de *workflows* de ETL em ambientes de *data warehousing* em nível conceitual.

Foram exploradas as práticas, técnicas e ferramentas utilizadas para construção de *workflows* de ETL, priorizando a modelagem conceitual e foram consideradas as seguintes questões: (i) a representação visual das operações

que compõem o processo de ETL pode facilitar a construção de ambientes de *data warehousing*, (ii) o uso de elementos gráficos que sejam simples e expressivos, pode contribuir para a abstração e para a visibilidade do tratamento necessário para compatibilização dos dados extraídos das fontes de dados operacionais com a estrutura proposta para o DW.

Assim surgiu a proposta de um novo modelo conceitual para *workflows* de ETL, denominado modelo Intuitive, que traz uma linguagem visual simples e expressiva, e que pode contribuir para facilitar a comunicação e o entendimento entre os projetistas e os usuários da área de negócios promovendo o engajamento da equipe na fase inicial da construção de ambientes de *data warehousing*. Além disso, vislumbramos que a padronização da modelagem conceitual com o modelo Intuitive sirva para documentar as decisões do projeto de construção de ETL e que essa documentação possa contribuir para facilitar a análise de impacto nas manutenções e a avaliação de cenários alternativos para melhorar a eficiência do processo. Embora o escopo da pesquisa tenha sido limitado a soluções de *data warehousing*, é possível que a aplicação do modelo Intuitive traga benefícios para outras soluções que envolvem sistemas de ETL como, por exemplo, soluções de mineração de dados.

## 1.4 **Organização do Trabalho**

A dissertação está organizada nos seguintes capítulos: o primeiro capítulo aqui descrito contém a Introdução, com o contexto, a motivação e os objetivos da pesquisa além da estruturação da dissertação; como parte do referencial teórico, no capítulo 2 são apresentados os conceitos de *data warehousing*, contemplando o ambiente, a construção e a arquitetura e, no capítulo 3, é apresentado o conceito de ETL como uma parte crítica do ambiente de *data warehousing*, contemplando as principais operações, as etapas e as ferramentas automatizadas para a construção de *workflows* de ETL; o capítulo 4 apresenta os trabalhos correlatos que abrangem a modelagem conceitual como parte da

construção de sistemas de ETL, descrevendo as principais abordagens do estado da arte; no capítulo 5 está descrita a proposta do modelo Intuitive, um novo modelo para a etapa de modelagem conceitual de *workflows* de ETL, contemplando os principais elementos, a representação gráfica e exemplos de aplicação; o capítulo 6 descreve como foi realizada a validação do modelo Intuitive e, no capítulo 7 são apresentadas as conclusões do trabalho e indicações para trabalhos futuros.

# Capítulo 2

# *DATA WAREHOUSE*

*Nesse capítulo são apresentados os conceitos fundamentais sobre ambientes de data warehousing, considerando as suas principais características, a construção, a arquitetura e a modelagem de dados. O entendimento desses conceitos é essencial para a compreensão da proposta de pesquisa deste trabalho.*

## 2.1 **Considerações Iniciais**

Soluções de *data warehouse* (DW) são usadas para atender a demanda de separar o ambiente operacional, constituído por aplicações que lidam com transações convencionais, do ambiente informacional, que provê suporte para análises e para a tomada de decisão estratégica (CIFERRI, 2002). O ambiente de *data warehousing* é complexo, sendo formado por elementos que compõem um modelo multidimensional para o armazenamento e o acesso aos dados. Nesse capítulo são descritos o processo de construção e os elementos da arquitetura de um ambiente de *data warehousing*.

## 2.2 **Construção do ambiente de *data warehousing***

A construção do ambiente de *data warehousing* é um grande processo contemplando diversas atividades que demandam conhecimentos tanto da área de negócios como também habilidades da área técnica. Golfarelli (2010) propõe que as atividades necessárias para a construção do DW sejam organizadas em etapas: (i) análise de requisitos, com a identificação das informações que são relevantes para apoiar o público alvo no processo de tomada de decisão, considerando as necessidades e a disponibilidade real de dados nas fontes operacionais, (ii) projeto conceitual, com a elaboração de um modelo visual expressivo e abstrato que seja independente da tecnologia; (iii) projeto lógico, que corresponde à criação de um modelo lógico a partir do modelo conceitual, que pode ser relacional (ROLAP), uma solução multidimensional (MOLAP) ou ainda, multidimensional híbrida HOLAP, (iv) construção do processo de ETL como um *workflow* de tarefas de transformação dos dados e mapeamentos necessários para carregar no DW os dados disponíveis nas fontes de dados operacionais, (v) projeto físico, que considera o uso de ferramentas e tecnologias nas tarefas de implementação e testes. De forma semelhante, na visão de Vassiliadis, Simitsis e Skiadopoulos (2002), a construção do ambiente de *data warehousing* envolve as seguintes atividades principais: (i) análise das fontes de dados e levantamento de requisitos, que tem como resultado a modelagem conceitual do DW, (ii) projeto lógico, que consiste na elaboração da estrutura de armazenamento do DW e definição das atividades de ETL (primeira etapa da construção do processo de ETL), (iii) projeto físico, contemplando a otimização e ajustes da estrutura proposta, com a definição dos índices para o DW e dos parâmetros para execução e (iv) construção (implementação e testes) dos sistemas de *software* necessários, implantação e estabelecimento das métricas

que são usadas pelos administradores. A Figura 1 ilustra, de forma abstrata, as principais etapas da construção do ambiente de *data warehousing*.

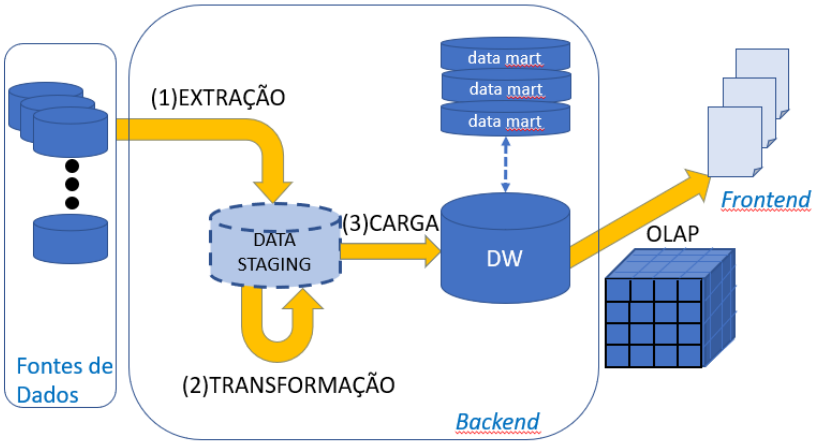



**Figura 1: Etapas da construção de um ambiente de data warehousing**

Fonte: Elaborada a autora

## 2.3 Arquitetura de um ambiente de *data warehousing*

A arquitetura de um ambiente de *data warehousing* é composta essencialmente pelo seguinte conjunto de elementos: fontes de dados (provedores de informação), *backend*, DW, OLAP e *frontend*. Esses elementos são ilustrados na Figura 2 e estão descritos nas próximas seções desse trabalho.

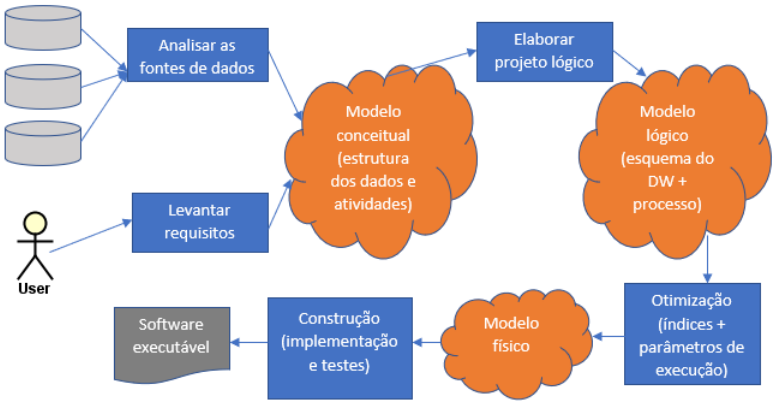



**Figura 2: Elementos básicos da arquitetura do ambiente de *data warehousing***

Fonte: Elaborada pela autora

### 2.3.1 **Fontes de dados**

Em um ambiente de *data warehousing*, as fontes de dados operacionais são usadas na fase mais inicial da construção do DW e apresenta como principais características (i) a autonomia, ou seja, cada fonte de dados possui independência e é usada de forma isolada para suporte a um determinado contexto de negócio; (ii) a heterogeneidade, indicando que as fontes de dados apresentam estruturas, modelos e semânticas variadas com dados não padronizados e, muitas vezes, não estruturados (por exemplo, arquivos formatados ou não, bancos de dados relacionais, bancos de dados objeto-relacionais, documentos html ou quaisquer outras formas de armazenamento) e (iii) a distribuição, pois as fontes de dados podem estar fisicamente localizadas em diferentes servidores que, inclusive, podem estar separados geograficamente.

### 2.3.2 **Camada *backend***

Em um ambiente de *data warehousing*, o *backend* é basicamente composto por (i) um repositório temporário de dados, chamado de *data staging area* (DSA) e por (ii) processos de ETL usados para extração dos dados dos diversos provedores, transformação desses dados e posterior carga no DW (KIMBALL; ROSS, 2002).

A área de *data staging* é usada para armazenar temporariamente os dados que foram extraídos dos provedores, para que possam ser acurados e transformados para posterior carga no DW. A transformação dos dados é complexa e sua execução pode ser demorada e computacionalmente cara. Com o armazenamento temporário no DSA, as operações de transformação aplicadas aos dados não interferem nas consultas analíticas complexas que são realizadas no DW. Além disso, é no *backend* que são aplicadas operações que envolvem algoritmos, ferramentas e processos complexos como ETL, carregamento,

atualização e expiração de dados (CIFERRI, 2002). O processo de ETL, que é detalhado no Capítulo 3 desse documento, corresponde às operações envolvidas na extração, limpeza, transformação e carga de dados. A extração de dados é a fase inicial em que é realizada a leitura dos dados das fontes de dados, o entendimento e o armazenamento desses dados no DSA que, posteriormente, serão tratados e carregados para a camada de DW. A fase de tratamento dos dados envolve uma série de manipulações dos dados como limpeza, padronização dos dados vindos de diferentes fontes e em vários formatos, eliminação de redundâncias e de duplicidades e integração de dados. A etapa final corresponde ao carregamento dos dados nos *data marts* presentes na camada de DW para que sejam indexados e posteriormente liberados para consultas dos usuários (KIMBALL; ROSS, 2002).

A operação de atualização dos dados tem o objetivo de manter a consistência e a relevância dos dados ao longo do tempo de forma que continuem sendo úteis para alimentar os sistemas de apoio à tomada de decisão. A atualização de dados pode ser realizada de forma periódica ou na ocorrência de um evento específico. Quanto à forma de atualização, há duas possibilidades: (i) a atualização pode ser realizada de forma incremental, na qual apenas os dados modificados nos provedores de informações, ou seja, apenas os novos dados (as diferenças nos dados), são refletidos no DW, ou (ii) pode ser feita a carga completa dos dados em substituição dos dados previamente carregados no DW.

A expiração dos dados corresponde à remoção dos dados do repositório para diminuição do volume armazenado. É aplicada em situações específicas, tais como quando o tempo de validade dos dados é atingido, ou quando os dados armazenados no DW não são mais necessários ou relevantes, ou ainda, quando não há espaço suficiente para o armazenamento de todos os dados (desde que o volume armazenado no DW é realmente muito grande ao longo do tempo).

### 2.3.3 *Data warehouse*

É a parte principal de um ambiente de *data warehousing*. Corresponde ao grande repositório de dados organizado de forma multidimensional para prover suporte, com eficiência, flexibilidade e escalabilidade, às consultas OLAP para apoio à tomada de decisão estratégica. Os dados armazenados no DW apresentam as seguintes principais características (CIFERRI, 2002): (i) os dados são orientados a assunto, por exemplo compras e vendas, os quais estão relacionados com elementos que são importantes ao contexto de negócio da organização para a qual foi construído, tais como cliente, produto, vendas e outros; (ii) os dados são integrados ou previamente tratados e acurados, com a redução das inconsistências e redundâncias existentes nos vários provedores de informações; (iii) os dados são não-voláteis, os seja, os dados permanecem estáveis por longos períodos, considerando que, sobre eles, são executados apenas duas grandes e complexas transações: carga (ou manutenção) dos dados e leitura (por meio das consultas OLAP) e (iv) históricos, o que significa que são relevantes durante algum período, diferente de dados operacionais, que são válidos no momento em que são acessados.

Os dados são organizados em níveis de agregação: (i) no nível inferior estão os dados detalhados obtidos do ambiente operacional e, portanto, com menor granularidade, (ii) nos níveis intermediários os dados são agregados de forma crescente e, (iii) no nível superior os dados são altamente agregados e resumidos. Assim, os dados são inicialmente carregados no nível inferior e passam por sucessivos níveis de agregação até atingir o nível superior. Periodicamente, como parte do processo de expiração de dados, os dados do nível inferior podem ser transferidos para um nível histórico, garantindo a manutenção do volume dos dados armazenados e a priorização dos dados mais recentes e mais relevantes. A granularidade dos dados armazenados impacta diretamente no volume dos dados e na flexibilidade oferecida para a realização de consultas: quanto mais detalhado, maior o volume e maior a flexibilidade para as consultas; quanto mais resumido, menor o volume de dados e menos flexíveis

são as consultas. Assim, a granularidade dos dados é uma decisão importante no projeto de um DW (CIFERRI, 2002). O DW pode ainda conter os *data marts* e o repositório de metadados. *Data marts* são subconjuntos dos dados contidos no DW, que podem ser usados como parte da construção do DW. Muitas vezes esses subconjuntos são separados por assunto e, geralmente, são específicos para departamentos ou unidades de uma organização. Os metadados descrevem como os dados são armazenados no DW e como foi realizado o processo de ETL, desde os dados de origem até o seu armazenamento (esquemas e procedência dos dados) e, portanto, permitem conhecer a estrutura e o significado dos dados armazenados (semântica) além de outras informações úteis para a administração dos dados.

### 2.3.4 **Consultas OLAP**

OLAP, *Online Analytical Processing*, é um conceito associado a interação do usuário com os dados. As consultas OLAP provêm a capacidade de análise dos dados por vários ângulos ou perspectivas, facilitando a manipulação, a descoberta de novos valores e a obtenção de conhecimento. As principais consultas OLAP são: *drill-down*, *roll-up*. *slice and dice*, *pivot* e *drill-across* (CIFERRI, 2002). Assim, o servidor OLAP, como parte da arquitetura do ambiente de *data warehousing*, provê visões multidimensionais dos dados armazenados no DW e nos *data marts*, usando o conceito do (hiper)cubo multidimensional de dados que caracteriza a modelagem de dados em camadas. É no servidor OLAP que são processadas as consultas analíticas sobre os dados armazenados no DW.

### 2.3.5 ***Frontend***

Na arquitetura de um ambiente de *data warehousing*, o *frontend* é composto por ferramentas que permitem a visualização dos resultados das

consultas analíticas e gerenciais realizadas sobre os dados armazenados no DW (CIFERRI, 2002), tais como (i) geradores de relatórios periódicos que permitem que os usuários realizem consultas mesmo sem conhecer os detalhes da estrutura de armazenamento dos dados, (ii) ferramentas estatísticas, que permitem consultas, análises estatísticas e a visualização dos dados em forma de gráficos, (iii) ferramentas OLAP, que permitem a realização de consultas sofisticadas com o uso de visões multidimensionais em um cubo de dados e a visualização dos dados considerando diversas perspectivas para a análise, (iv) ferramentas de mineração de dados, usadas para explorar informações, padrões e tendências de negócio que podem estar ocultos nos dados presentes no DW.

Uma característica importante das ferramentas do *frontend* é a forma de apresentação dos resultados obtidos nas consultas, pois devem permitir a total visualização das informações, padrões e agregações.

## 2.4 **Modelagem dimensional**

A modelagem dimensional dos dados tem o objetivo de permitir a consulta e a análise dos dados a partir de diversas perspectivas (visões do usuário), também chamadas dimensões. Essa forma de modelagem prioriza a agregação dos dados. Os diferentes níveis de agregação de dados facilitam a realização de consultas e análises de tendências cujos resultados são subsídios para decisões estratégicas. A seguir são apresentados conceitos relacionados com a modelagem dimensional tais como aspectos estáticos, incluindo o cubo de dados, e os aspectos dinâmicos que correspondem às operações OLAP frequentemente aplicadas em DW (BRITO, 2018; CIFERRI, 2002).

### 2.4.1 **Aspectos *estáticos* da modelagem dimensional**

Os aspectos estáticos da modelagem dimensional envolvem os conceitos de medidas numéricas, fatos e dimensões, representados. Dimensão representa uma perspectiva (visão) dos dados armazenados no DW. Um fato está relacionado a um assunto tratado pelo DW. A medida numérica é o valor quantitativo que representa o fato e pode ser definida como uma função das dimensões correspondentes (valor no espaço multidimensional). Nesse sentido, as dimensões definem o contexto para um fato que, por sua vez, pode ser quantificado por uma medida numérica.

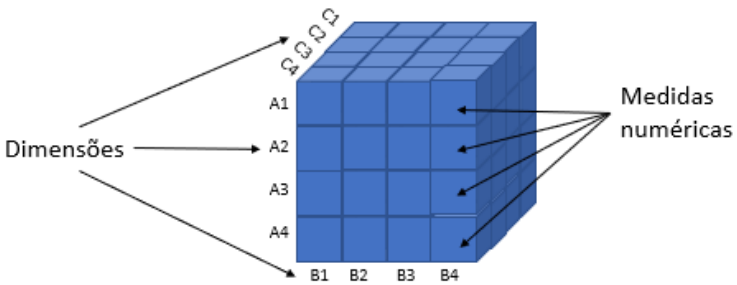



**Figura 3: Modelagem dimensional – cubo de dados**

Fonte: Adaptado de CIFERRI (2002)

A dimensão é composta por atributos. Os atributos de uma dimensão podem ser relacionados em hierarquia. A hierarquia de atributos especifica os níveis de agregação e a granularidade dos itens de dados. A representação gráfica para a modelagem estática de DW é um cubo multidimensional (ilustrado na Figura 3), onde cada face corresponde a uma dimensão e cada célula é um fato que tem uma medida numérica associada. A estrutura de armazenamento dos dados no DW é mais facilmente entendida com a representação do cubo de dados.

### 2.4.2 **Aspectos dinâmicos da modelagem dimensional**

Os aspectos dinâmicos da modelagem de um DW correspondem às operações OLAP realizadas sobre os dados do DW. As operações são (i) *drill-down,* que permite a análise mais detalhada dos dados considerando os vários níveis de agregação seguindo para a menor granularidade, (ii) *roll-up*, é o inverso do *drill-down*, ou seja, permite a visualização mais resumida dos dados considerando os vários níveis de agregação no sentido da maior granularidade (maior agregação); (iii) *slice and dice*, que restringe o conjunto de dados considerando um valor específico (*slice*) ou uma faixa de valores (*dice*), (iv) *pivot*, que reorienta a visão multidimensional dos dados, permitindo analisa-los em diferentes perspectivas e, (v) *drill-across*, que permite comparar medidas numéricas relacionadas de acordo com as dimensões em comum. Esse conjunto de operações básicas corresponde à visão dinâmica sobre os dados do DW e possibilita que os usuários realizem consultas complexas e análises sobre os dados.

## 2.5 **Considerações Finais**

Nesse capítulo foram apesentados os conceitos essenciais para entendimento de um ambiente de *data warehousing* detalhando as atividades relacionadas à construção, os elementos da arquitetura e a modelagem dimensional. O próximo capítulo tem foco no processo de ETL em ambientes de *data warehousing*, detalhando os principais elementos, bem como as práticas, técnicas e ferramentas aplicadas na construção e operação.

# Capítulo 3

# ETL EM DW

*Nesse capítulo é apresentado o processo de ETL que é uma parte crítica do ambiente de data warehousing, com as grandes atividades de extração, transformação e carga; são apresentadas também, as principais etapas da construção sistemas de ETL, contemplando a modelagem conceitual e a modelagem lógica. Essas informações foram essenciais para a realização desse trabalho. Ao final há um panorama das ferramentas automatizadas utilizadas para construção de ETL em DW.*

## 3.1 Considerações Iniciais

Em um ambiente de *data warehousing*, o processo de ETL tem grande importância e é bastante complexo, sendo que a implementação exige um grande esforço do desenvolvedor e a execução consome bastante tempo e recursos computacionais. Esse processo é tradicionalmente responsável pela extração (ou obtenção) dos dados de fontes distintas e heterogêneas, a transformação dos dados originais, com o processamento, análise, limpeza, integração e padronização dos dados e, finalmente, o armazenamento dos dados no DW (BALA; BOUSSAID; ALIMAZIGH, 2015).

Assim, ETL é a camada de *software* existente entre as fontes de dados e o DW, sendo implementada como uma sequência de operações, geralmente na forma de um *workflow* de processos, na qual os dados são iterativamente tratados e acurados até serem adequadamente integrados, e armazenados no

DW. No *workflow* de ETL, entre as fontes de dados e o destino, pode haver vários ramos ou subfluxos de processos. Não há limite para a quantidade de operações em um *workflow* de ETL – cenários complexos podem exigir muitas operações de transformação e de limpeza de dados – e pode haver vários ramos concorrentes.

Uma operação de ETL é uma abstração de um componente de código que executa uma ou mais tarefas sobre os dados a fim de compatibiliza-los com a estrutura proposta para o DW, tais como: manipulação ou transformação dos dados, inicialização de atributos, agregações e outros tratamentos ou regras do domínio do negócio. Há várias formas para a implementação das operações de ETL: comandos SQL, funções definidas pelos próprios desenvolvedores usando alguma linguagem de programação ou ainda com o reuso de componentes predefinidos. Qualquer que seja a forma escolhida para a implementação das operações, a eficiência do processo de ETL depende da experiência e das habilidades do projetista no trabalho de definição do *workflow* (ALI; WREMBEL, 2017).

## 3.2 **Atividades do processo ETL**

O processo de ETL é tradicionalmente composto por 3 atividades principais: começa com (i) a **extração** de dados dos diversos provedores heterogêneos e distribuídos, (ii) a segunda atividade corresponde à **transformação,** limpeza e integração dos dados e (iii) a terceira é a atividade de **carga** dos dados no DW (PAN; ZHANG; QIN, 2018). Essas atividades, que são executadas na camada de *backend* do ambiente de *data warehousing,* estão ilustradas na Figura 4. Para apoiar a atividade de transformação usa-se o DSA, onde os dados ainda não estão necessariamente armazenados segundo a modelagem dimensional e podem, inclusive, ser armazenados em tabelas no modelo relacional ou em arquivos, da melhor forma possível para que os objetivos da atividade de transformação sejam alcançados. Algumas soluções

de suporte a decisão podem, inclusive, acessar os dados diretamente no DSA para a criação de relatórios ou para a mineração de dados e, para essas soluções o processo de ETL é tão importante quando para os ambientes de *data warehousing*.

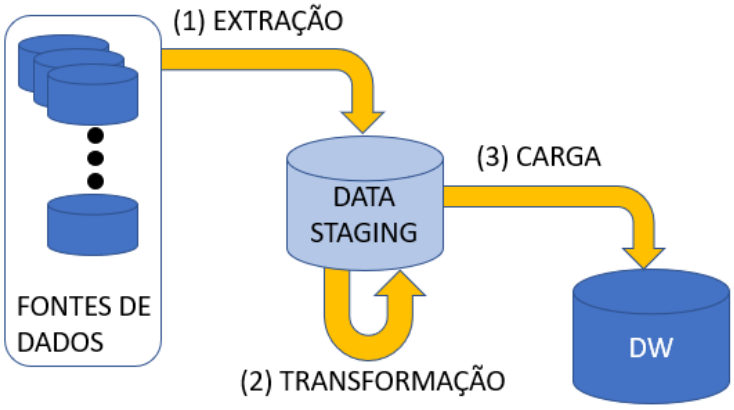



**Figura 4: Principais atividades do processo de ETL: extração, transformação e carga.**

Fonte: Elaborado pela autora

Comparando com as atividades de extração e de carga, a atividade de transformação geralmente demanda maior tempo para implementação porque costuma ser exclusiva (*ad hoc*) para cada cenário. A execução da atividade transformação também demanda muito tempo porque, caso aconteça algum problema durante a execução dessa etapa ou caso ocorra alguma mudança em alguma regra da transformação, toda a atividade deve ser executada novamente e isso pode encarecer o processo como um todo (PAN; ZHANG; QIN, 2018). As atividades de extração, transformação e carga em um ambiente tradicional de *data warehousing,* são descritas nas próximas seções.

## Extração

A primeira importante atividade de ETL é a extração de dados brutos a partir das fontes de dados para posterior transformação e carga no DW. A implementação da atividade de extração inclui a conexão e o acesso às fontes de dados que têm diferentes formatos e podem estar armazenadas localmente ou em diversos servidores. Para a carga inicial dos dados no DW, todos os dados das fontes de dados são extraídos e encaminhados para as demais atividades

de ETL. Quanto maior for o volume de dados, mais tempo será consumido para execução da atividade de extração. Após a carga inicial, a extração costuma ser feita de forma incremental, com o objetivo de manter os dados do DW atualizados. Assim, a atividade de extração se encarrega de identificar, nas fontes de dados, as alterações que ocorreram desde a extração anterior (por exemplo, novos dados inseridos ou dados alterados) e, então, apenas essas diferenças são efetivamente extraídas e encaminhadas para o DSA para a atividade de transformação dos dados.

Dois pontos importantes que devem ser considerados para a extração de dados são: (i) os sistemas que alimentam as fontes de dados não devem sofrer intervenções ou as intervenções devem ser mínimas e, (ii) a extração não pode impactar negativamente a execução e o desempenho dos demais sistemas envolvidos, ou seja, deve ser não intrusiva e, por isso, a extração é executada preferencialmente durante uma “janela de manutenção”, período em que os sistemas envolvidos, em geral, ficam mais ociosos.

Dependendo do volume de dados, a carga completa de dados pode demandar muito tempo para execução e, geralmente, a extração incremental consome apenas uma fração desse tempo. Para a extração incremental, é necessário monitorar as fontes de dados para a identificação das diferenças e as estratégias empregadas nesse monitoramento são denominadas *Change Data Capture* (SIMITSIS; VASSILIADIS; SELLIS, 2005). Algumas técnicas de CDC são apresentadas a seguir .

### ***Change Data Capture*** **(CDC)**

Além da carga inicial e completa dos dados, em um ambiente de *data warehousing* o processo de ETL é usado para manter o conteúdo do DW atualizado com relação às alterações ocorridas nas fontes de dados. Nesse contexto, na extração de dados são usadas estratégias de CDC para monitorar

as modificações ocorridas nas fontes de dados (inclusões, alterações e remoções de dados) e identificar as diferenças entre o estado corrente das fontes de dados e o estado que havia na extração anterior. As soluções de CDC são importantes para a manutenção do DW pois, ao permitir que apenas os dados modificados sejam efetivamente extraídos, representa uma redução significativa no tempo necessário para a execução da extração. Basicamente, as estratégias de CDC podem ser divididas nas seguintes categorias:

- CDC baseado em *logs*. Muitas vezes, os sistemas que alimentam as fontes de dados registram as transações em arquivos de *log*. O CDC baseado em *logs*, percorre esses arquivos para identificar as diferenças que devem ser extraídas para atualização do DW.
- Utilização da coluna de auditoria. Pode haver fontes de dados que, ao invés de registrar as transações em arquivos, marcam os registros alterados com um *timestamp* na coluna de auditoria. O CDC compara essa informação com a data e o horário da extração de dados mais recente. As diferenças, ou seja, os itens de dados onde a data de alteração é mais recente que a data da extração anterior, são extraídas para posterior atualização do DW. Uma limitação dessa técnica de CDC é que não é possível identificar dados que tenham sido removidos da fonte de dados.
- Comparação de *snapshots*. Um *snapshot* é como uma foto do conteúdo dos dados em um certo momento, e mostra o valor dos dados armazenados, ou seja, o estado corrente dos dados. Nesse caso, a atividade de extração armazena um *snapshot* da fonte de dados no momento da execução. Após a extração dos dados, é possível que o estado da fonte de dados seja modificado com a inclusão de novos dados, alterações ou remoções, gerando um novo estado que será, então, capturado quando a atividade de extração for novamente executada. O CDC usa algoritmos para calcular as diferenças entre os dois *snapshots*, ou seja, para identificar as diferenças entre o estado corrente da fonte de dados e o estado da fonte de dados na extração anterior. As diferenças identificadas resultam no arquivo delta que é usado para a atualização incremental do DW.

- *Triggers*. O recurso de *trigger* presente nos Sistemas Gerenciadores de Banco de Dados (SGBDs) relacionais pode ser usado para disparar o CDC quando alguma alteração é aplicada na fonte de dados. No entanto, essa técnica só se aplica a dados estruturados por exemplo, segundo o modelo relacional, e pode demandar intervenções nos sistemas que alimentam as fontes de dados e pode onerar a execução desses sistemas. Assim, triggers não costumam ser usados como estratégia de CDC na atividade de extração do processo de ETL.

A decisão sobre qual a melhor estratégia de CDC para a extração de dados no processo de ETL leva em consideração os sistemas que alimentam das fontes de dados e a estrutura desses dados (estruturados ou não estruturados). De acordo com Simitsis, Vassiliadis e Sellis (2005), soluções de CDC que fazem a comparação de *snapshots* são as mais comumente usadas para ETL em ambientes de *data warehousing*.

**Transformação**

A segunda grande atividade do processo de ETL corresponde à transformação dos dados. Os dados brutos, que foram inicialmente extraídos dos diversos provedores de informação, passam pela transformação para, posteriormente, serem armazenados no DW. A transformação envolve tarefas para limpeza, conversão, padronização e integração dos dados e, de forma geral, é a etapa que consome mais tempo para implementação e, também, para a execução. Basicamente, o objetivo da transformação é compatibilizar os dados brutos extraídos das diversas fontes de dados com o esquema proposto para os *data marts* e o DW. Para não interferir e nem provocar impactos negativos nas fontes de dados e nos sistemas que as alimentam, geralmente as tarefas de transformação são executadas na DSA do ambiente de *data warehousing.*

A transformação dos dados lida com a resolução de conflitos e de problemas tanto em nível de esquema, ou seja, a compatibilidade entre os esquemas dos diversos provedores de informação e o esquema do DW, como

também em nível de instância de dados. No nível de esquema podem ocorrer conflitos tais como: (i) sinônimos: quando um mesmo elemento é representado por nomes diferentes nas fontes de dados (por exemplo: em uma fonte de dados, usa-se “cliente” para representar as pessoas que realizam compras, enquanto em outra fonte usa-se “comprador”); (ii) homônimos: quando um único termo é usado para representar diferentes elementos (por exemplo, em uma fonte de dados usa-se “nome” para representar os nomes dos alunos, enquanto em outra usa-se “nome” para representar os nomes das disciplinas oferecidas); (iii) conflito semântico: quando um mesmo elemento apresenta diferentes conceitos (exemplo: usa-se “pessoa” para representar os clientes cadastrados e, em outra fonte, para representar qualquer tipo de usuário quer sejam clientes, vendedores ou fornecedores). No nível de instância, alguns conflitos que podem ser encontrados são: (i) registros duplicados, (ii) diferentes níveis de granularidade (exemplo: vendas por dia ou vendas por mês) e, (iii) incompatibilidade no valor de atributos (valores diferentes para o mesmo dado em diferentes fontes de dados). Para todos os conflitos ou problemas de compatibilidade entre as fontes de dados e o destino, é necessário implementar mapeamentos e funções que permitam tratá-los (SIMITSIS; VASSILIADIS; SELLIS, 2005).

A limpeza de dados, como parte da atividade de transformação, corresponde a uma série de tarefas para a modificação ou a eliminação de dados indesejáveis tais como redundâncias, duplicidades, inconsistências, valores inválidos, entre outros, com o objetivo de melhorar a qualidade dos dados. Vassiliadis (2009) ressalta que a limpeza constitui uma parte importante do processo de ETL e que poderia até ser considerada como uma atividade adicional, separada da transformação dos dados. O desafio da limpeza dos dados está relacionado com a identificação de dados que representam um mesmo acontecimento ou objeto do mundo real e que apresentam valores diferentes, sendo que a maior dificuldade está associada a dados do tipo texto porque, na maioria das vezes, os usuários podem fornecer quaisquer valores para esses atributos sem qualquer padronização ou qualquer filtro. Assim, a identificação de duplicação de dados envolve algoritmos de comparação que consideram a similaridade entre os dados e calcula a medida da “distância” entre

esses dados. Normalmente, o processo de limpeza dos dados é construído de forma específica para cada cenário porque é necessário considerar as características e particularidades do conjunto de dados que se deseja tratar.

**Carga**

A atividade de carga do processo de ETL corresponde ao armazenamento dos dados no DW. Os dados de entrada para e a atividade de carga foram previamente extraídos das diversas fontes de dados e passaram por tarefas de limpeza, padronização, conversão e integração que foram executadas na atividade de transformação. Portanto, os dados usados para a entrada da atividade de carga, foram previamente tratados e acurados.

A carga de dados envolve desafios técnicos que devem ser resolvidos pelo projetista tais como: (i) usar uma sequência de inclusões para tratar cada item de dado pode ser um processo muito lento dependendo do volume de dados, além de envolver riscos de ocorrências de falhas durante o processamento e, assim, a realização de carga em massa com alguma ferramenta específica de SGBDRs para processamento em *batch* costuma ter melhor desempenho e ser menos sujeito a falhas; (ii) em situações de grande volume de dados de entrada, para conseguir diferenciar os dados novos, ou seja, aqueles que ainda não foram carregados para o DW, daqueles que foram previamente carregados e que sofreram alguma modificação desde a carga de dados anterior e, o uso de *open-loop-fetch* para carregar cada item para o DW pode apresentar problemas de desempenho; assim, usa-se preferencialmente o recurso de *merge* de dados que é provido pelos SGBDs; (iii) ainda sobre estratégias que podem ser usadas para a carga de dados, o projetista deve considerar a presença de índices e de visões materializadas pois essas estruturas podem degradar o tempo de execução da carga dos dados para o DW. De modo geral, a implementação das tarefas de carga de dados no processo de ETL deve considerar a melhoria do desempenho e a maximização do uso de processamento paralelo e distribuído (SIMITSIS; VASSILIADIS; SELLIS, 2005).

## 3.3 **Construção do processo de ETL**

De forma geral, os projetos de soluções tecnológicas e de soluções de banco de dados apresentam algum nível de abstração. O princípio de abstração em um projeto corresponde à omissão de detalhes técnicos enquanto se dá destaque a elementos essenciais ao entendimento do problema que é tratado e da solução que é proposta para o problema. Cada nível de abstração pode servir a um diferente propósito no contexto de um projeto: nas fases iniciais, quando o conhecimento sobre o problema é limitado, uma abstração de alto nível pode contribuir para uma visão geral do assunto que se deseja tratar e conforme o conhecimento sobre o problema se torna mais claro, mais detalhes técnicos podem ser adicionados permitindo obter níveis mais detalhados de abstração. Modelos são usados para representar diferentes níveis de abstração, com o uso de conceitos e de operações. De acordo com os tipos de conceitos que são oferecidos, um modelo pode ser classificado como (i) conceitual, com alto nível de abstração e que, portanto, representa uma visão mais facilmente entendida por usuários não técnicos, (ii) lógico, contendo mais detalhes que aproximam da visão da equipe técnica que atua no projeto e (iii) físico, com informações técnicas suficientes que contribuem para as decisões tecnológicas do projeto (ELMASRI; NAVATHE, 2011).

Um projeto de construção de ETL envolve, basicamente, as seguintes etapas: (i) modelagem conceitual, (ii) modelagem lógica, (iii) projeto físico e implementação e (iv) otimização. As etapas de modelagem conceitual e de modelagem lógica são descritas a seguir. As demais etapas não estão no escopo desse trabalho de pesquisa.

**Modelagem conceitual de ETL**

A modelagem conceitual é o processo de documentar formalmente um problema ou o domínio de um problema para o entendimento e a integração das partes interessadas em um projeto que será desenvolvido, contribuindo para

resolver o problema descrito. De forma geral, o modelo conceitual complementa a descrição dos requisitos dos usuários e, ao final do desenvolvimento, ajuda a avaliar se o produto esperado foi desenvolvido (MOODY, 2005). Segundo Mehmood, Cherfi e Comyn-Wattiaum (2009), a modelagem conceitual tem os objetivos de (i) contribuir na descoberta dos requisitos dos usuários, (ii) representar formalmente a realidade observada e (iii) ser a base para implementar e manter o sistema. Portanto, os modelos conceituais são usados para apoiar o desenvolvimento e para a manutenção de sistemas. Gemino e Wand (2004) afirmam que um modelo conceitual é formado por construtores (símbolos gráficos) e regras para a combinação desses construtores; o conjunto de construtores e regras de um modelo conceitual formam a gramática desse modelo. O resultado da aplicação da gramática para a representação de uma abstração é um diagrama que é chamado de esquema conceitual.

Na construção de ETL, a modelagem conceitual é realizada no início do projeto, quando ainda não há detalhes técnicos sobre a solução que será desenvolvida: são analisados os requisitos dos usuários, as estruturas das fontes de dados e a estrutura proposta para o armazenamento dos dados no DW. Nessa fase, um modelo de alto nível de abstração é usado para representar, de forma concisa e visual, os requisitos dos usuários do DW, as características das fontes de dados, as características da estrutura proposta para o armazenamento dos dados no DW e as transformações que devem ser aplicadas a esses dados (KOUGKA et al., 2017). O principal artefato gerado na modelagem conceitual de ETL é o esquema conceitual. O esquema conceitual apresenta o relacionamento entre os elementos dos provedores de dados e os elementos da estrutura proposta para o DW (VASSILIADIS; SIMITSIS; SKIADOPOULOS, 2002). Assim, o esquema conceitual é um recurso precioso para (i) a documentação das decisões tomadas na construção de ETL, (ii) a análise de impacto das manutenções necessárias para atendimento de demandas que ocorrem no ciclo de vida do DW, tais como alterações nas fontes de dados, evolução dos requisitos ou das regras de negócio, necessidade de melhoria no desempenho das consultas, correção de erros cometidos durante a fase de

projeto (EL-SAPPAGH et al., 2011) e (iii) explorar cenários alternativos para buscar soluções mais eficientes para o processo de ETL.

De forma geral, a modelagem conceitual na construção de ETL tem o objetivo de propor e organizar as várias tarefas, funções e mapeamentos necessários para compatibilizar os dados brutos das fontes de dados com a estrutura proposta para o DW e, assim, atender as necessidades de informações dos usuários do nível gerencial. O modelo conceitual de ETL precisa (i) ser facilmente compreendido pelos usuários finais que são conhecedores do negócio e que não necessariamente possuem conhecimento sobre as tecnologias e conceitos técnicos subjacentes a um ambiente de data warehousing, (ii) conter informações necessárias para a construção e a manutenção, contribuindo para reduzir o esforço e o retrabalho.

Algumas abordagens têm sido propostas para facilitar, formalizar e padronizar a modelagem conceitual na construção de ETL e entre essas abordagens estão os modelos baseados em grafos, em UML (*Unified Modelling Language*), ontologia e BPMN (*Business Process Model Notation*), além de . Essas abordagens são detalhadas no capítulo 4.

**Características de qualidade para modelagem conceitual de ETL**

De acordo com El-Sappagh et al. (2011), a modelagem conceitual de ETL deve apresentar características tais como simplicidade, completude e possibilidade de customização. Outras características importantes são clareza, consistência e não ambiguidade. Adicionalmente, o modelo conceitual de ETL precisa (i) ser facilmente entendido pelos usuários finais que são conhecedores do negócio e que muitas vezes não conhecem profundamente a tecnologia e (ii) contribuir para diminuir o esforço dos projetistas e desenvolvedores durante a construção do processo de ETL.

De acordo com Gemino e Wand (2004), a gramática de um modelo conceitual deve ter como principais características a expressividade, a completude e a simplicidade. Os autores defendem também que a avaliação da qualidade de um modelo conceitual pode ter foco na avaliação da qualidade dos diagramas ou na avaliação da qualidade da gramática do modelo. Moody (2005) considera que não há uma forma padronizada de avaliar a qualidade de modelos conceituais. O modelo conceitual deve ser compreensível pelas partes interessadas: usuários da área do domínio do negócio, projetistas e desenvolvedores.

É desejável que haja um equilíbrio entre poder expressivo, completude e simplicidade. Em um modelo conceitual, o poder expressivo corresponde à representatividade semântica dos construtores providos pela gramática e dos diagramas conceituais gerados. A completude pode ser entendida como sendo a medida de quanto a gramática é suficiente para representar uma abstração do domínio de negócio que está sendo tratado. Já a simplicidade pode ser medida como a quantidade de construtores e regras providas pela gramática (quanto menos construtores e regras, mais simples é o modelo) ou a quantidade de elementos usados em um determinado diagrama (quanto menos elementos, mais simples é o diagrama). É importante notar que a simplicidade do modelo pode impactar no seu poder expressivo, ou seja, é possível que modelos mais simples não sejam tão expressivos e, por isso, podem ser mais difíceis de serem compreendidos. Outro ponto importante na avaliação do modelo conceitual é o grau de conhecimento do público-alvo: modelos mais simples podem ser mais facilmente entendidos por usuários não técnicos enquanto modelos conceituais mais complexos podem prover detalhes que ajudem nas atividades dos desenvolvedores e dos projetistas. Com relação à não ambiguidade, é desejável que um diagrama gráfico seja disponibilizado, evitando que existam diferentes interpretações.

Considerando esses fatores, nesse trabalho definimos um conjunto de características intragramaticais que nos ajudaram na comparação entre os

modelos conceituais de ETL encontrados na literatura e, também, nos ajudaram a definir características importantes na definição do Modelo Intuitive.

**Modelagem lógica de ETL**

Após a etapa de modelagem conceitual, é necessária a evolução (ou conversão) para a modelagem lógica. A modelagem lógica do processo de ETL é usada para descrever os detalhes técnicos das tarefas envolvidas, tais como algoritmos e especificidades, incluindo as possíveis dependências entre as tarefas, descrever os algoritmos de operações e funções específicas do domínio de negócio, e estabelecer as regras e as restrições que devem ser aplicadas na execução do processo. O objetivo dessa etapa é organizar a sequência de execução das tarefas do *workflow*, desde as fontes até o armazenamento dos dados no DW. Além disso, a modelagem lógica permite definir o tratamento necessário para retomada da execução do processo no caso falhas ou interrupções (recuperabilidade). Os principais modelos lógicos de processo de ETL são baseados em grafos: Grafo de Arquitetura ou Grafo Acíclico Direcionado Parametrizado (ALI; WREMBEL, 2017).

A modelagem lógica baseada em Grafo de Arquitetura é usada na construção de ETL como complemento à modelagem conceitual baseada em grafos (ALI; WREMBEL, 2017; VASSILIADIS; SIMITSIS; SKIADOPOULOS, 2002). Nessa abordagem, a semântica de cada operação de ETL é representada com o uso de dois elementos principais: (i) nós, que representam as operações com os atributos correspondentes e (ii) ligações (conexões), que representam os relacionamentos entre as operações. Os relacionamentos entre as operações podem ser dos tipos: (i) relacionamento *regulador*, que indica o uso de uma fonte de dados externa para popular um atributo, (ii) relacionamento *fornecedor*, que indica o fluxo que parte de uma fonte de dados e segue um registro de destino, (iii) relacionamento *parte de*, que representa o relacionamento entre os atributos e uma operação, um conjunto de registros ou uma função, (iv) relacionamento *instância de*, que representa a relação entre tipos de dados e atributos. É possível ter diferentes níveis de detalhes nos esquemas lógicos baseados em

Grafo de Arquitetura: no nível inicial, por exemplo, são representados basicamente a origem dos dados, o destino e as operações de transformação e de limpeza que são aplicadas aos dados.

Na modelagem lógica com Grafo Acíclico Dirigido Parametrizado (DAG-P), dois elementos são usados: (i) vértices do grafo, representando as operações, as transformações e as bases de dados e (ii) linhas, representando o fluxo dos dados desde a fonte até o destino; além disso, está prevista a inclusão de anotações para descrever parâmetros, métricas de qualidade ou requisitos de negócio, que podem ser necessários para a execução do processo de ETL (ALI; WREMBEL, 2017, WILKINSON, et al., 2010). Na Figura 5 é apresentado um exemplo de esquema lógico com o modelo de Grafo de Arquitetura e na figura 6 é apresentado um exemplo de esquema lógico com DAG-P.

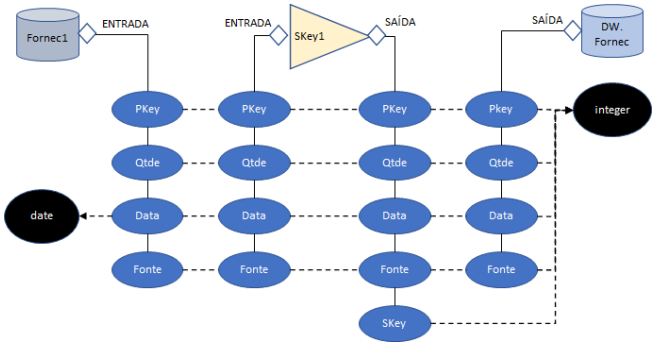



**Figura 5: Exemplo de esquema lógico de ETL com Grafo de Arquitetura**

Fonte: Adaptada de VASSILIADIS, SIMITSIS e SKIADOPOULOS (2002).

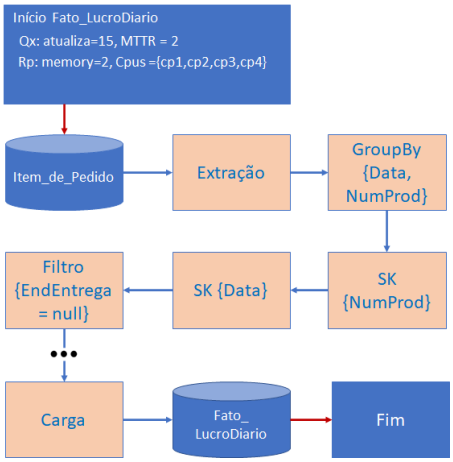



**Figura 6: Exemplo de esquema lógico de ETL com DAG-P**

Fonte: Adaptada WILKINSON et al. (2010).

Após a modelagem lógica, a construção do processo de ETL evolui para a etapa de projeto físico e, nesse ponto, aplica-se, principalmente, a linguagem XML que é suportada por ferramentas automatizadas tais como o Pentaho Data Integration. Uma visão geral de ferramentas automatizadas que são usadas para a construção de ETL é apresentada a seguir.

## 3.4 **Ferramentas de ETL**

A necessidade crescente de obtenção de informações úteis em tempo hábil a partir do gigantesco volume de dados que é gerado atualmente, tem motivado a realização de pesquisas e o desenvolvimento de ferramentas automatizadas com os objetivos de facilitar e de agilizar a construção do processo de ETL. Nesse contexto, as ferramentas automatizadas consideram (i) a criticidade do processo de ETL para a manutenção dos dados em um ambiente de *data warehousing*, (ii) o esforço necessário, por parte dos desenvolvedores e projetistas, para a implementação de processos de ETL eficientes, (iii) o alto consumo de recursos computacionais e de tempo para a execução desse processo e (iv) o enorme volume de dados que precisam ser tratados em tempo hábil.

O ARKTOS (VASSILIADIS et al., 2001) é uma das primeiras ferramentas propostas para a construção de ETL no contexto de *data warehousing.* Trata-se de um *framework* para projeto e implementação de processos de ETL que envolve uma interface gráfica e linguagens declarativas que permitem determinar operações que são comuns em ETL tais como transformações e limpeza, priorizando o tratamento aplicado aos atributos desde as fontes de dados até o armazenamento no DW. O PygramETL foi proposto por Thomsen e Pedersen (2009), e trata-se de uma ferramenta baseada na linguagem Python, que não possui uma interface gráfica, e que defende o uso de recursos de programação para a construção de soluções de ETL de forma mais rápida. Liu, Thomsen e Pedersen (2011) propuseram o *framework* ETLMR como uma

estratégia para desenvolvimento de ETL, que usa como base o modelo BPMN e transforma esse modelo em código executável. Embora essas abordagens tragam benefícios para a construção de ETL, até onde pudemos avaliar, ainda não há consenso ou adoção maciça de uma ferramenta única.

Além das abordagens citadas anteriormente, várias ferramentas de ETL estão disponíveis no mercado. Muitas dessas ferramentas são proprietárias e há, também, opções de ferramentas de ETL *open source* tais como: Pentaho Data Integrator (também conhecida como Kettle), Talend Open Studio e CloverETL, que são bastante poderosas e permitem a construção de processos complexos. De forma geral, as ferramentas de ETL facilitam o trabalho dos desenvolvedores e projetistas provendo ambientes gráficos e componentes predefinidos que possibilitam a construção visual do *workflow* de ETL sem a necessidade de escrever código-fonte em qualquer linguagem.

- Pentaho Data Integration (www.pentaho.com/product/dataintegration) é bastante usada para construção de ETL em ambientes de *data warehousing*; foi desenvolvido em JAVA e tem um ambiente gráfico para a criação de *workflows* de ETL, sem a necessidade de escrever linhas de código. A ferramenta provê acesso a um conjunto de componentes (metadados) que podem ser reutilizados e que facilitam e agilizam a construção de ETL. Além disso, é compatível com vários formatos de entrada e de saída de dados tais como planilhas, arquivos texto, SGBDs. A linguagem gráfica é convertida em código XML.
- Talend Open Studio (www.talend.com) é uma ferramenta *open source* desenvolvida pela empresa Talend, que corresponde a um gerador de código-fonte para a implementação de *scripts* com encapsulamento de código JAVA. Oferece um ambiente gráfico que permite modelar visualmente o *workflow* de ETL, sem a necessidade de escrever linhas de código. O Talend provê um conjunto de elementos básicos para a construção das operações frequentemente presentes em *workflows* de ETL, tais como conversão, combinação e atualização dos dados; esses elementos podem ser reutilizados, contribuindo para a redução do esforço necessário para a

construção e manutenção de sistemas de ETL (CHAKRABORTY; PADKI; BANSAL, 2017).

- CloverETL (www.cloveretl.com) é uma ferramenta *open source* para a construção de ETL em DW. Foi desenvolvida em JAVA e apresenta uma interface gráfica com componentes para a representação visual do *workflow* de ETL. Os fluxos de dados são representados como grafos onde as setas representam as dependências entre as operações desde as fontes de dados, passando pelas transformações, até o destino que pode ser um *data mart* ou o DW. Os grafos são transformados em código XML e, além dos componentes pré-definidos, o CloverETL permite que o desenvolvedor construa novos componentes e os aplique na definição do *workflow* de ETL. É possível combinar grafos para compor cenários de ETL complexos. (CHAKRABORTY; PADKI; BANSAL, 2017).

Além das ferramentas de ETL apresentadas nessa seção, várias outras estão disponíveis, tais como, Oracle Warehouse Builder, IBM Infosphere, MSSQLServer Integration Services e Informatica PowerCenter for Enterprise Data Integration.

Por fim, as ferramentas automatizadas são especialmente necessárias para projetos complexos que envolvem grandes volumes de dados, provendo recursos para a otimização do processo de ETL. Além disso, contribuem para que a atualização e a manutenção dos dados contidos nos *data marts* e no DW sejam feitas com mais eficiência. Não há uma padronização dos elementos gráficos e dos recursos providos pelas diversas ferramentas de ETL (KABIRI; CHIADIMI; 2013). De forma geral, essas ferramentas provêm um ambiente gráfico e têm foco nas etapas de modelagem lógica e de projeto físico, quando os desenvolvedores e projetistas já conhecem os detalhes técnicos do projeto de DW; o projeto físico sucede as etapas de modelagem conceitual e de modelagem lógica.

## 3.5 **ETL ou ELT**

Trabalhos recentes têm explorado alternativas para o processo de ETL. Uma linha de estudos compara o processo tradicional de ETL com a proposta de ELT (*extract – load – transform*). Na abordagem ELT, após extrair os dados brutos das diversas fontes de dados, esses são carregados diretamente para a DSA, onde a transformação é realizada sob demanda. Assim, a execução das tarefas de transformação, integração e limpeza dos dados é adiada até o último momento possível, ou seja, até que os dados acurados sejam necessários para compor o resultado de alguma consulta do usuário. Em outras palavras, os dados são tratados e acurados somente quando forem efetivamente solicitados. Essa abordagem pode ser especialmente interessante em cenários onde os requisitos de negócio mudam com frequência (MUKHERJEE; KAR, 2017).

Além da abordagem ELT, alternativas têm sido propostas para melhorar o desempenho e a flexibilidade do processo de ETL. De acordo com Pan, Zhang e Qin (2018), a estratégia ECL-TL propõe a divisão do processo tradicional de ETL em duas partes: ECL (*extract - clean - load*) e TL (*transform - load*), que são ligadas pela *Middle Library*. O componente ECL é usado para limpar os dados brutos extraídos das diversas fontes de dados e para alimentar a *Middle Library*, que pode ser entendida como um DW intermediário, realizando o tratamento de dados inconsistentes, valores inválidos ou dados incompletos; a *Middle Library* serve de entrada para o componente TL, que transforma os dados e os carrega no DW para posteriormente servir aos sistemas de apoio à decisão. As vantagens dessa abordagem são: (i) melhor estabilidade porque, como a etapa de transformação é isolada das etapas de extração e carga dos dados, problemas de extração ou de limpeza não impactam a transformação; (ii) como os dados contidos na *Middle Library* estão limpos e integrados, é possível compartilha-los com tarefas de mineração de dados; (iii) flexibilidade, porque é possível alterar as regras de transformação sem alterar a extração e a carga dos dados; (iv) redução do tráfico de rede e do consumo de recursos computacionais.

## 3.6 **Considerações finais**

O processo de ETL é parte essencial de um ambiente de *data warehousing* e é crítico para o tratamento e atualização dos dados. Sistemas de ETL são grandes e complexos e demandam grande esforço, tempo e conhecimentos especializados, para construção e a manutenção. Vários trabalhos de pesquisa e de desenvolvimento relacionados a esse tema têm sido feitos nas últimas décadas e, apesar desse esforço, ainda não há um consenso ou uma recomendação sobre as melhores práticas a serem aplicadas na construção de sistemas de ETL. Além disso, os desafios atualmente impostos pelos grandes volumes de dados gerados em altíssima velocidade motivam inovações nessa área, onde ainda há espaço para contribuições e melhorias.

# Capítulo 4

# TRABALHOS RELACIONADOS

*Nesse capítulo são descritas e analisadas as principais soluções existentes na literatura para a modelagem conceitual do processo de ETL, que foram estudadas nesse trabalho de pesquisa e que foram muito relevantes para a produção dessa dissertação de Mestrado.*

## 4.1 **Considerações iniciais**

Esse capítulo aborda trabalhos relacionados ao tema desta pesquisa de Mestrado que tratam especificamente da modelagem conceitual de *workflows* de ETL em ambientes de *data warehousing*, a saber: modelo conceitual de ETL com grafos, modelo conceitual de ETL com UML, uso de ontologias para modelagem conceitual de ETL, modelo conceitual de ETL com BPMN e o modelo Kantara. Essas abordagens visam facilitar, formalizar e padronizar a construção de *workflows* de ETL e, segundo nosso conhecimento, são muito relevantes.

## 4.2 **Modelo conceitual com grafos**

A modelagem conceitual com grafos está entre as primeiras que foram propostas para apoio à construção de *workflows* de ETL (VASSILIADIS; SIMITSIS; SKIADOPOULOS, 2002). Prioriza o mapeamento dos atributos das

diversas fontes de dados compatibilizando-os com os atributos da estrutura proposta para armazenamento no DW. Basicamente, essa abordagem propõe a organização do processo em camadas: (i) camada de meta-modelo, que corresponde a um conjunto de estruturas genéricas para a representação das várias operações do processo e, (ii) camada de *template,* que é um subconjunto dos elementos da camada de meta-modelo e que são customizados para representar as operações frequentemente usadas na construção de ETL. Assim, a representação do cenário de ETL utiliza instâncias das estruturas da camada de meta-modelo (VASSILIADIS; SIMITSIS; SKIADOPOULOS, 2002; ALI; WREMBEL, 2017).

A modelagem com grafos envolve uma notação visual com elementos que representam os atributos e as transformações necessárias para esses atributos, permitindo representar graficamente o mapeamento dos atributos de entrada (das fontes de dados) para o formato proposto para o destino (estrutura do DW). O trabalho de modelagem envolve, então, (i) a identificação das estruturas dos provedores de dados (entrada) e da estrutura proposta para o DW (saída composta por fatos e dimensões), (ii) a identificação da fonte de dados principal e de outras fontes candidatas, (iii) a identificação das transformações e descrição das regras necessárias para o mapeamento dos atributos desde a fonte principal até o destino e (iv) a adição de comentários para descrever restrições relativas à execução do processo.

De acordo com Ali e Wrembel (2017), essa abordagem tem algumas limitações: (i) a notação gráfica proposta não favorece a compreensão dos diagramas gerados, (ii) contempla apenas dados estruturados e (ii) depende das habilidades e decisões do projetista para a construção de um processo de ETL eficiente. A Figura 7 é um exemplo de esquema conceitual elaborado usando grafos.

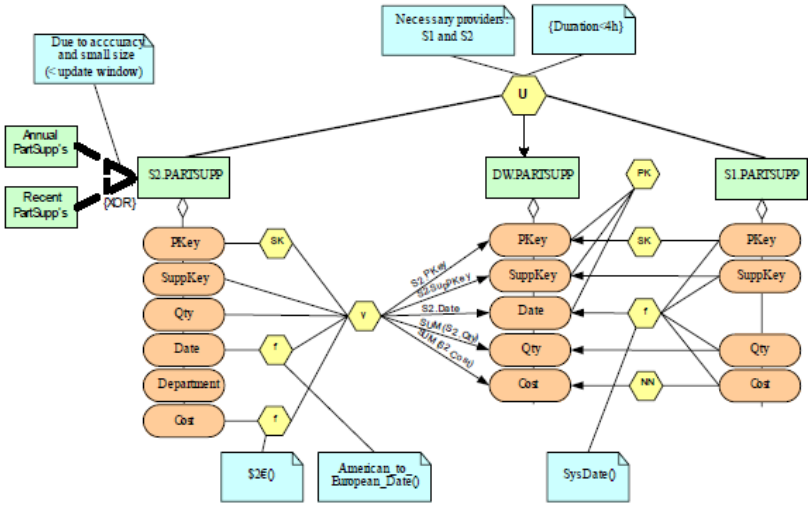



**Figura 7: Exemplo de uso do modelo conceitual com grafos para um cenário de ETL**

Fonte: Extraída de VASSILIADIS, SIMITSIS e SKIADOPOULOS (2002)

## 4.3 **Modelo conceitual com UML**

A UML surgiu no início dos anos 1990, no contexto da área de pesquisa em Engenharia de *Software* (ES), especialmente voltada para o paradigma de programação orientada a objetos (POO). Define elementos que são usados para compor diversos diagramas, tais como diagrama de casos de uso, diagrama de classes, diagrama de sequência, diagrama de atividades, dentre outros. Na ES, os diagramas da UML são usados para representar, de forma padronizada, as visões estática e comportamental de um produto de *software* que será desenvolvido, considerando tanto o ponto de vista do usuário do produto como, também, o ponto de vista do desenvolvedor.

A abordagem proposta por Trujillo e Luján-Mora (2003) para a modelagem conceitual de ETL envolve a utilização do Diagrama de Classes da UML ao qual são adicionados ícones que representam abstrações de operações de ETL, tais como agregação, filtro, criação de chaves artificiais, *merge*, *join*, dentre outras. Além disso, para processos de ETL grandes e complexos, a abordagem envolve o uso do Diagrama de Pacotes da UML para abstração e representação do fluxo

dos dados entre as operações. A Figura 8, extraída de Trujillo e Luján-Mora (2003), tem um exemplo de esquema conceitual de ETL com o uso da UML, compreendendo a operação de *merge*.

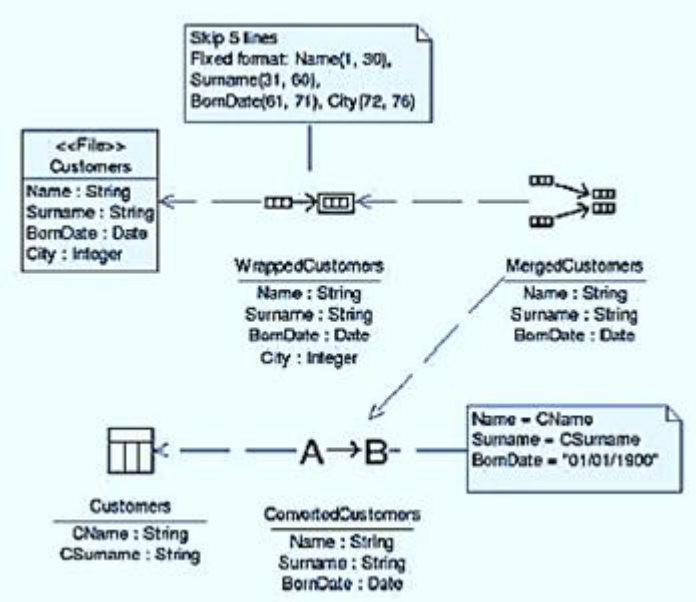



**Figura 8: Exemplo de uso da UML para um cenário de ETL**

Fonte: Extraída de TRUJILLO e LUJÁN-MORA (2003)

Considerando que a UML é uma linguagem formal amplamente conhecida, a adoção dessa forma de modelagem não exige novos conhecimentos dos desenvolvedores, o que pode facilitar sua adoção. Porém, de acordo com Ali e Wrembel (2017), embora a modelagem com UML seja uma forma menos complexa que a modelagem com grafos, há algumas limitações: (i) essa proposta contempla apenas o tratamento de dados estruturados (dados não estruturados precisam ser convertidos) e (ii) a eficiência do processo de ETL depende das habilidades e decisões do projetista para a modelagem e a construção do processo. Ademais, o uso da linguagem UML não garante o tratamento de características intrínsecas do processo de ETL, por exemplo, por meio da identificação de operações que são frequentemente presentes de *workflows* de ETL, causando perda semântica na modelagem de cenários de ETL. O esquema resultante, por conter muita informação, é visualmente poluído

e, por isso, traz uma visão segmentada do *workflow* e não favorece a visão do fluxo dos dados completo, desde a origem até o destino.

## 4.4 **Modelo conceitual com Ontologia**

O uso de Ontologia possibilita que a modelagem conceitual de ETL seja feita de forma semiautomática, facilitando o tratamento das transformações, mapeamentos e conflitos entre os dados de entrada (das fontes de dados) e a estrutura de saída (DW) e, também, o tratamento de problemas relacionados à semântica de estruturas heterogêneas (ALI; WREMBEL, 2017).

Uma ontologia é uma forma de representar o significado e os relacionamentos de um domínio de negócio com o estabelecimento de um conjunto de termos comuns, ou seja, de um vocabulário controlado. No contexto de DW, o conceito de ontologia pode ser usado para inferência dos mapeamentos necessários entre os dados heterogêneos dos provedores e a estrutura do DW (SKOUTAS; SIMITSIS, 2007). Basicamente, os passos envolvidos na modelagem conceitual de ETL com ontologia são (i) estabelecimento de um vocabulário comum, com a definição do domínio da aplicação e dos requisitos associados ao DW que se deseja construir, (ii) representação das fontes de dados com base no vocabulário que foi estabelecido, (iii) construção de uma ontologia para a aplicação que está sendo analisada. O vocabulário comum é definido pelo projetista a partir das informações relacionadas com a área de negócio do DW que se deseja construir e são adicionadas anotações sobre as fontes de dados tais como conceitos envolvidos no *workflow*, conjunto de atributos considerando os tipos e formatos, domínio dos valores permitidos para os atributos. Assim, a ontologia da aplicação é formada pelo vocabulário e as anotações que detalham as fontes de dados. A Figura 9, extraída de Simitsis et al. (2010) retrata um esquema conceitual de ETL elaborado com ontologia.

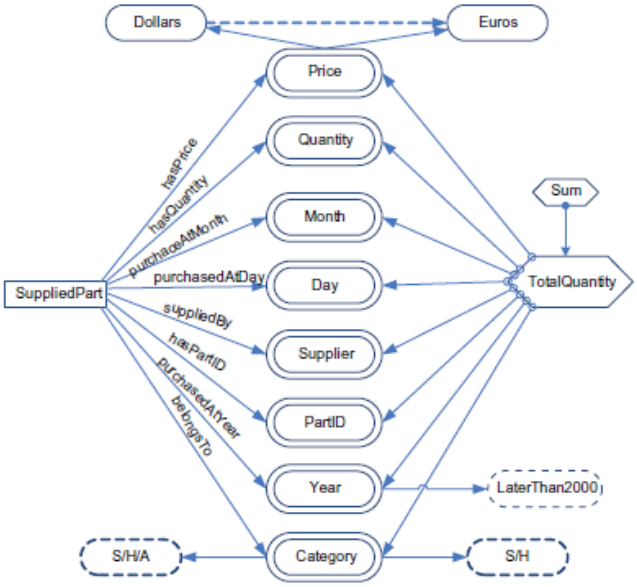



**Figura 9: Exemplo de uso de ontologia para modelagem conceitual de um cenário de ETL**

Fonte: Extraída de SIMITSIS et al. (2010)

De acordo com Ali e Wrembel (2017), a modelagem baseada em ontologia é eficiente para o tratamento de dados estruturados; para dados semiestruturados ou não-estruturados foi proposta a utilização de grafos de armazenamento de dados (DSG) que permitem a representação do modelo conceitual de forma padronizada e que utilizam a ontologia que foi inicialmente definida. Porém, o uso de ontologias não atende plenamente a representação do fluxo de dados em *workflows* de ETL: o uso de ontologias é um recurso importante para o mapeamento dos atributos em um dado contexto de negócio e pode ser considerado como um recurso adicional, mas falha na representação do fluxo dos dados que é a parte mais importante na modelagem do *workflow* de ETL. Além disso, um ponto desfavorável do uso dessa abordagem é a dependência da preexistência de uma ontologia para a área de negócio que é alvo do projeto de DW. Por fim, a ontologia resolve, basicamente, conflitos sintáticos e estruturais de dados heterogêneos, mas ainda necessita de intervenção humana para resolver conflitos semânticos que ocorrem na integração de dados.

## 4.5 **Modelo conceitual com BPMN**

A proposta de utilização da notação do BPMN (*Business Process Model and Notation*) para representação do modelo conceitual de ETL é independente de plataforma e apresenta operadores específicos para representar as tarefas do processo de ETL, tais como *gateways*, eventos, conectores e artefatos. Após a representação do diagrama BPMN, o modelo é transformado com a Linguagem de Execução de Processo de Negócio (BPEL). Dessa forma, a modelagem possibilita o uso de diversas ferramentas e a adequação aos vários requisitos da aplicação de DW que será construída (ALI; WREMBEL, 2017). Adicionalmente, o modelo conceitual baseado em BPMN pode ser melhorado utilizando uma metodologia em camadas que permite, por meio de sucessivas iterações, partir dos requisitos da aplicação, seguir para o modelo lógico e, dele, para a implementação física, considerando medidas quantitativas e qualitativas, tais como tratamento de recuperação de falhas e desempenho, entre outras (SIMITSIS et al., 2009). Além disso, elementos adicionais de BPMN podem ser utilizados para a construção de um *workflow* de ETL mais eficiente e com alta qualidade. Porém, de forma similar às limitações descritas para o uso da linguagem UML, o modelo BPMN por ser genérico não garante o tratamento de características intrínsecas do processo de ETL, por exemplo, por meio da identificação das principais operações realizadas em *workflows* de ETL. Ou seja, não há elementos criados especificamente para tratar dessas características intrínsecas e com isso há perda semântica na modelagem de cenários de ETL.

## 4.6 **O modelo Kantara**

O modelo Kantara foi proposto por Kabiri, Wadjinny e Chiadmi (2011) e trata-se de um *framework* para modelagem de processos de ETL. Na modelagem conceitual, o Kantara permite representar graficamente o processo de ETL em alto nível de abstração. A notação gráfica tem 6 componentes: componente de extração, componente de transformação e

componente de carga, ligações entre as atividades, anotações para descrever quaisquer elementos no diagrama e parâmetros para descrever detalhes sobre o ambiente operacional. Esse trabalho foi complementado por Kabiri, Wadjinny e Chiadmi (2012) com a proposta de uma forma de organização das atividades de construção do processo de ETL. Assim, a notação gráfica é bastante simples e carece de melhor expressividade.

## 4.7 **Considerações finais**

Nesse capítulo foram apresentadas as abordagens propostas na literatura para realizar a modelagem conceitual de *workflows* de ETL. Apesar dos esforços já realizados, ainda não há um consenso sobre a abordagem mais apropriada e eficiente para aplicação na etapa de modelagem conceitual de *workflows* de ETL e, por não haver processo padronizado, há deficiência no estabelecimento de metas e métricas para os projetos, dificultando a comparação da eficiência entre os projetos e a promoção da melhoria contínua.

Ali e Wrembel (2017) definem características que podem ser usadas para análise e classificação das abordagens de modelagem conceitual de ETL descritas nesse capítulo: (i) nível de automação, que corresponde ao grau de dependência das decisões e intervenções do projetista para a aplicação da abordagem na construção de ETL, (ii) formato das fontes de dados, ou seja, o nível de estruturação dos dados de entrada (estruturado, semiestruturado ou não estruturado) que é priorizado pela abordagem, (iii) possibilidade de uso de funções pré-definidas (reusabilidade), (iv) tratamento e priorização de métricas de qualidade para direcionar a definição da modelagem e (v) padronização dos elementos de modelagem, ao permitir o uso de um *framework* unificado para a conversão do modelo conceitual para o modelo lógico e do modelo lógico para a implementação física do processo de ETL.

A tabela 1 sintetiza as características dos modelos citados nesse capítulo. As características usadas na comparação são descritas no capítulo 7.

| | **Grafo** | **UML** | **Ontologia** | **BPMN** | **Kantara** | **Intuitive** |
|---|---|---|---|---|---|---|
| **Característica principal** | Representação visual do tratamento aplicado aos atributos desde as fontes até o DW | Adaptação da linguagem unificada para representar as dependências entre as atividades de ETL | Representação do significado e dos relacionamentos do domínio do negócio | Adaptação da representação visual de processos de negócio para representar processos de ETL | Simplicidade da notação gráfica | Representação visual dos elementos do *workflow* |
| **Construtores específicos para ETL** | X | | | | X | X |
| **Construtores reusáveis** | X | X | | X | X | X |
| **Construtores padronizados** | X | X | X | X | X | X |
| **Completude** | X (com o modelo lógico) | X | | X | | X |
| **Poder expressivo** | | | | | | X |
| **Simplicidade da gramática** | | | X | | X | |
| **Simplicidade dos diagramas** | | | | | X | X |
| **Extensibilidade: adição de novas funções específicas** | X | X | | | X | X |

**Tabela 1. Resumo das características das modelos conceituais de ETL**
Fonte: elaborado pela autora

A partir do estudo das abordagens descritas, consideramos que o uso de grafos consiste no estado da arte na modelagem conceitual de *workflows* de ETL, por reunir características importantes para modelagem conceitual de ETL, tais como: (i) ter sido desenvolvida especificamente para o tratamento das características de *workflows* de ETL, abrangendo componentes reutilizáveis que permitem representar as operações que são frequentemente aplicadas aos dados, (ii) admitir funções criadas pelos usuários, (iii) possuir um conjunto de elementos padronizados e (iv) apresentar os recursos necessários para prover suporte à etapa de modelagem conceitual do processo de ETL.

# Capítulo 5

# O MODELO INTUITIVE

*Nesse capítulo é apresentado o modelo Intuitive, criado especificamente para apoiar a etapa de modelagem conceitual de workflows de ETL em ambientes de data warehousing. O modelo Intuitive define um conjunto de conceitos, os operadores, que permitem representar, em alto nível de abstração, as operações que são aplicadas aos dados em workflows de ETL, bem como, a organização dessas operações e os repositórios de dados. Por ser independente de tecnologia e por usar notação visual simples e expressiva, o modelo Intuitive contribui para o engajamento dos usuários não técnicos desde a fase inicial do projeto, por exemplo usuários da área de negócio, colaborando para a correta captura dos requisitos e para a obtenção de resultados que agregam valor ao negócio e à organização.*

## 5.1 **Considerações iniciais**

Em projetos de *data warehousing*, a construção e a manutenção do processo de ETL exigem considerável esforço e conhecimentos técnicos. A comunidade de pesquisa tem contribuído com novas abordagens para a construção de *workflows* de ETL, mas, segundo Ali e Wrembel (2017), essas abordagens ainda exigem significativo esforço dos projetistas, demandam que os usuários de negócio tenham conhecimentos técnicos suficientes para entender e validar os projetos e, além disso, ainda não há um padrão que seja

amplamente aceito e adotado. Assim, é desejável ter uma abordagem para modelagem conceitual de *workflows* de ETL que seja unificada e padronizada, que contribua para facilitar a construção, a validação e a manutenção dos *workflows* de ETL, alcançando assim os objetivos de qualidade.

Nesse contexto é proposto o modelo Intuitive, que é voltado para a modelagem conceitual de *workflows* de ETL. O modelo Intuitive possui um conjunto de conceitos chamados operadores, que representam operações de ETL, relacionamentos entre essas operações (fluxos) e, também, repositórios de dados. Os operadores têm notação gráfica simples e expressiva e servem para agilizar o trabalho das fases iniciais do projeto de DW, contribuindo para o engajamento dos usuários não técnicos, por exemplo usuários da área de negócios, e a equipe técnica envolvida em projetos de ETL. O modelo Intuitive é detalhado nas próximas seções.

## 5.2 **Operadores**

O modelo Intuitive possui um conjunto de conceitos com alto nível de abstração, denominados de operadores. Os operadores são usados para representar operações de ETL, relacionamentos entre essas operações (fluxos) e, também, áreas de armazenamentos de dados, ou seja, repositórios. Para representar um *workflow* de ETL, os operadores são combinados entre si com o uso de setas unidirecionais que indicam a propagação dos dados desde as fontes até o destino. O início de um *workflow* de ETL sempre é um ou mais repositórios que representam as fontes de dados e, de forma semelhante, o final de um *workflow* de ETL sempre é um ou mais repositórios, sendo que o principal é o DW. Assim, os operadores constituem uma linguagem visual padronizada que permite a representação abstrata das sucessivas operações de extração, transformação e carga que são aplicadas aos dados no processo de ETL.

Cada operador tem uma notação gráfica específica, sendo definido por entrada, parâmetros e saída. A entrada pode ser (i) unária, ou seja, permite apenas um conjunto de dados, (ii) binária, com dois conjuntos de dados, ou ainda (iii) n-ária, com vários conjuntos de dados (dois ou mais conjuntos de dados). Os parâmetros podem ser (i) nomes de atributos dos conjuntos de dados de entrada ou de saída (exemplos: funcNome, funcMatricula), ou (ii) condições que são especificadas por meio de expressões relacionais (funcCidade = São Carlos) ou expressões lógicas (exemplo: funcEstadoSigla = SP AND funcMatricula > 32879), (iii) critérios que indicam ordenação crescente (asc) ou decrescente (desc), (iv) indicação da precedência para o tratamento dos dados (exemplo: A – B), ou ainda (v) lista de atribuições (exemplo: funcEstadoSigla ← SP, funcSexo ← M). As operações relacionais ( =, >, <, <>, >=, <=) e as operações lógicas (NOT, AND, OR) seguem as definições já consagradas em linguagens de programação, na álgebra relacional e na linguagem SQL. Além disso, a saída, que é um conjunto de dados que pode ser direcionado para repositórios ou para outras operações de ETL, pode ser (i) unária, ou seja, apenas um conjunto de dados, (ii) binária, com dois conjuntos de dados, ou (iii) n-ária, com vários conjuntos de dados (dois ou mais conjuntos de dados).

Os operadores do modelo Intuitive são classificados em categorias, considerando as características e os efeitos que causam sobre os dados ou sobre a organização do processo, a saber: (i) operadores de armazenamento, ou seja, repositórios, (ii) operadores de manipulação de dados, (iii) operadores de agregação, (iv) operadores de inicialização, (v) operadores de fluxo (ou seja, que lidam com processos), e (vi) operadores especiais que tratam de especificidades e que complementam as funcionalidade providas pelos demais operadores.

Os operadores propostos são detalhados nas próximas seções com a descrição de sua funcionalidade, seguida pela descrição das entradas, dos parâmetros e das saídas (resultados). Também é apresentado um exemplo da aplicação de cada operador.

### 5.2.1 Operadores de armazenamento

Os operadores de armazenamento representam áreas de armazenamento de dados, tais como repositórios, arquivos, planilhas ou bases de dados. Um operador de armazenamento pode (i) indicar o ponto de início de um *workflow* de ETL representando uma fonte de dados e, nesse caso, não há entradas, (ii) pode receber dados resultantes da aplicação de algum outro operador e, nesse caso, a entrada é unária. De forma semelhante, um operador de armazenamento pode ser (i) o ponto final de um *workflow* de ETL e, nesse caso, não há saídas ou (ii) pode ter uma saída unária, quando os dados são direcionados para algum outro operador. Não é permitido direcionar dados diretamente de um operador de armazenamento para outro operador de armazenamento, ou seja, em um *workflow* de ETL, entre dois operadores de armazenamento sempre é necessário ter pelo menos um outro operador que não seja de armazenamento. Os operadores de armazenamento propostos são DataWarehouse, DataMart, DataLake, DataSet, TempDataSet, FailDataSet.

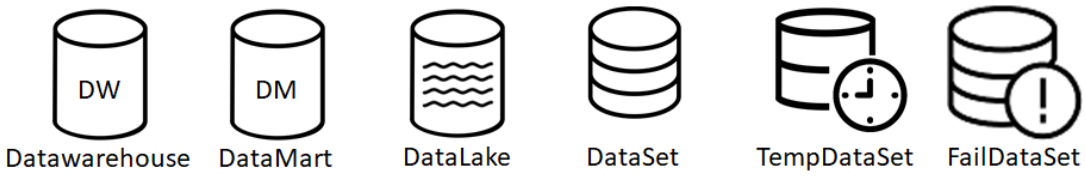



**Figura 10: Notação gráfica dos operadores de armazenamento**

Fonte: Elaborada pela autora

A Figura 10 ilustra os operadores de armazenamento de dados que são descritos a seguir.

#### Operador DataWarehouse

O operador DataWarehouse representa o grande repositório onde os dados tratados ao longo de todo o processo de ETL são carregados e armazenados para as consultas gerenciais: os dados do DW são resultantes da aplicação de sucessivas operações de extração, transformação e carga de dados. De forma

geral, o DW é o ponto final do processo de ETL, onde os dados ficam disponíveis para as consultas OLAP; alternativamente, em alguns ambientes de *data warehousing*, os dados do DW podem ser ainda direcionados para formar visões materializadas ou *data marts* usados para as consultas gerenciais. Assim, na representação conceitual do *workflow* de ETL, a entrada e a saída para o operador DataWarehouse são ambas unárias e não obrigatórias.

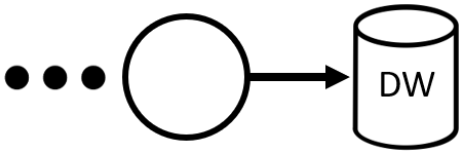



**Figura 11: Cenário de uso do operador DataWarehouse**

Fonte: Elaborado pela autora

A Figura 11 representa o uso do operador DataWarehouse com uma entrada que é um conjunto de dados resultante de alguma operação no *workflow* de ETL e, como nenhuma saída está representada, então o operador DataWarehouse é o ponto final de um *workflow* e os dados poderiam ser usados para consultas gerenciais.

### Operador DataMart

Em um ambiente de *data warehousing*, *data marts* são subconjuntos de dados do DW que são separados por assunto, por exemplo, dados de departamentos ou de unidades de uma organização; essa separação de dados pode preceder o armazenamento dos dados no DW (abordagem *bottom-up* de construção de um ambiente de *data warehousing*) ou, em alguns ambientes, os dados do DW podem ser direcionados para construir *data marts* que são usados para as consultas gerenciais (abordagem *top-down* de construção de um ambiente de *data warehousing*) . Assim, na representação conceitual do *workflow* de ETL, a entrada e a saída do operador DataMart são ambas unárias e não obrigatórias.

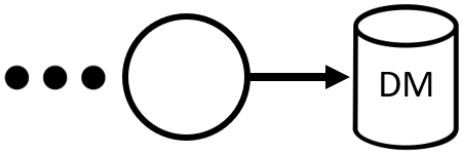



**Figura 12: Cenário de uso do operador DataMart**

Fonte: Elaborado pela autora

A Figura 12 ilustra o uso do operador DataMart com um conjunto de dados de entrada resultante de uma operação de ETL, possivelmente alguma agregação e, nesse caso, como nenhuma saída está representada, os dados contidos no *data mart* serão consumidos por consultas gerenciais.

### Operador DataLake

O operador DataLake corresponde a uma área de armazenamento convencionalmente chamada *data lake*, que contém um grande volume de dados brutos em diferentes formatos: dados estruturados, estruturados e semiestruturados. De forma geral, os dados de um *data lake* são processados apenas quando a informação é necessária para responder alguma consulta OLAP e, nesse sentido, pode servir como fonte para ferramentas de análise e consulta de dados. Em outro contexto, o *data lake* pode servir como um DSA para posterior carga de dados no DW. Assim, a entrada e a saída do operador DataLake são ambas unárias.

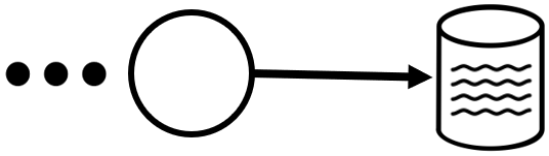


**Figura 13: Cenário de uso do operador DataLake**

Fonte: Elaborado pela autora

A Figura 13 ilustra o uso do DataLake com um conjunto de dados de entrada resultante de uma operação de ETL e, nesse caso, como nenhuma saída está representada então possivelmente os dados contidos no *data lake* serão consumidos por ferramentas de análise e consulta de dados fora do escopo do processo de ETL.

### Operador DataSet

O operador DataSet representa uma área de armazenamento de dados tal como repositórios, arquivos, planilhas ou bases de dados. Pode ser usado como ponto inicial ou como ponto final do *workflow* ou ainda como uma área de armazenamento intermediária como, por exemplo, para representar uma parte do DSA.

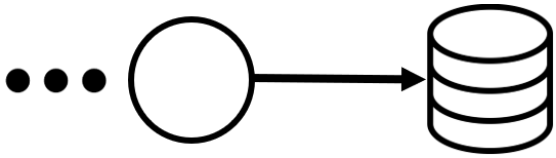


**Figura 14: Cenário de uso do operador DataSet**

Fonte: Elaborado pela autora

A Figura 14 apresenta um possível cenário de uso do operador DataSet onde um conjunto de dados resultante de alguma operação é armazenado e, nesse exemplo, é o ponto final do processo.

### Operador TempDataset

O operador TempDataSet corresponde a um caso particular do operador DataSet, que representa uma área temporária de armazenamento de dados. Uma situação para aplicação do TempDataSet é a necessidade de *backup* de dados ou a representação de um conjunto de dados de apoio para o processo que será logo em seguida descartado.

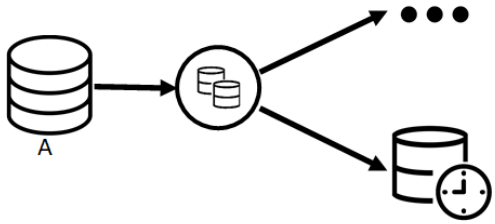



**Figura 15: Cenário de uso do operador TempDataSet**

Fonte: Elaborado pela autora

A Figura 15 apresenta um possível cenário de uso do operador TempDataSet onde o conjunto de dados resultante de um operador Copy é direcionado para o operador TempDataSet.

**Operador FailDataset**

O operador FailDataSet é um caso particular do operador DataSet e representa uma área de armazenamento de dados que podem ter sido rejeitados por uma operação ou um *log* de execução, como um ponto de término do *workflow* de ETL.

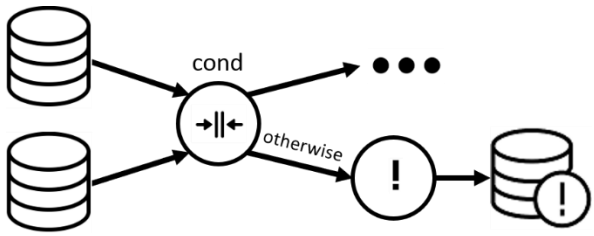



**Figura 16: Cenário de uso do operador FailDataSet**

Fonte: Elaborado pela autora

A Figura 16 apresenta um possível cenário de uso do operador FailDataSet onde, na aplicação do operador Join, os dados que não atendem à condição estabelecida são direcionados ao operador Fail, indicando um caminho alternativo que é finalizado com o armazenamento dos dados no FailDataSet.

## 5.2.2 Operadores de manipulação de dados

Os operadores de manipulação de dados são usados para representar as tarefas de transformação e de limpeza que são aplicadas aos dados extraídos das diversas fontes para torná-los compatíveis com a estrutura proposta para o DW: Filter, Union, Split, Join, Diff, Intersect, Sort, Update e Copy.


**Figura 17: Notação gráfica dos operadores de manipulação de dados**

Fonte: Elaborado pela autora

Os símbolos gráficos usados para representar os operadores de manipulação de dados são apresentados na Figura 17 e as descrições são apresentadas a seguir.

### Operador Filter

O operador Filter é usado para a seleção de subconjuntos de dados de acordo com condições estabelecidas. A entrada para o operador Filter é unária e a saída é n-ária. Como parâmetros, para cada saída do operador Filter deve ser estabelecida uma condição de seleção. O operador Filter não altera o esquema dos dados, ou seja, o seu formato composto por uma ou mais listas de atributos.

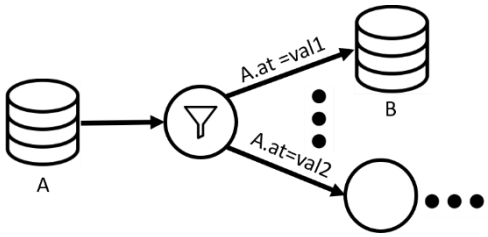



**Figura 18: Cenário de uso do operador Filter**

Fonte: Elaborado pela autora

Na Figura 18 a notação gráfica do operador Filter é apresentada, com um conjunto de dados de entrada, A, as condições A.at = val1 e A.at = val2 e duas saídas: um conjunto de dados é direcionado para o repositório B, e um segundo conjunto de dados é direcionado para alguma outra operação do *workflow*. Supondo que A contém dados de Cliente, que A.at corresponde a Região, val1 é Sul e val2 é Norte, o repositório B irá conter os dados de Clientes da Região Sul e os dados dos Clientes da Região Norte são direcionados para alguma operação específica.

**Operador Union**

O operador Union é usado para juntar os itens de dados de dois conjuntos fornecidos. Assim, a entrada para o Union é comumente binária, mas pode ainda ser n-ária, composta por conjuntos de dados que possuem obrigatoriamente o mesmo esquema. A saída é unária, contendo todos os itens do primeiro conjunto com a adição de todos os itens dos demais conjuntos, sem duplicidades, porque os itens repetidos são eliminados. O operador Union não altera o esquema dos dados. Além disso, fica implícito que para que a união dos dados seja iniciada, é necessário que os conjuntos de dados de entrada estejam preparados, ou seja, as tarefas anteriores aplicadas aos dados devem estar terminadas para o início da operação de união. Há, portanto, a garantia de sincronismo entre os diversos conjuntos de dados de entrada.

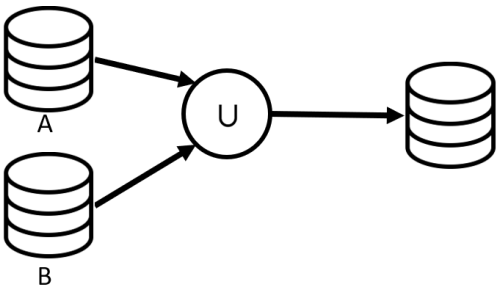



**Figura 19: Cenário de uso do operador Union.**

Fonte: Elaborado pela autora

A notação gráfica do operador Union é apresentada na Figura 19, com dois conjuntos de dados de entrada, A e B, e um único conjunto de dados de saída, contendo os dados de A aos quais são adicionados os dados de B sem duplicidades, e o conjunto de dados resultante é armazenado em um repositório. Supondo que A contém dados de Clientes da Região Sul e que B contém dados de Clientes da Região Norte, o resultado da aplicação do operador Union é o conjunto dos dados de Clientes das duas regiões Sul e Norte.

Uma situação em que o operador Union é comumente aplicado é a atualização dos dados do DW, onde dados novos e tratados são unidos aos dados previamente carregados no DW em um processamento anterior; em um outro exemplo, o operador Union pode ser aplicado sobre os dados originalmente

armazenados em duas fontes de dados distintas e que possuem o mesmo esquema; o resultado é um conjunto de dados contendo todos os itens dos dois conjuntos de entrada, sem duplicidades.

De certa maneira, o operador Union tem o efeito “inverso” do operador Filter. Enquanto o operador Filter tem o efeito de separar os itens de um conjunto de dados de entrada dando origem a dois ou mais subconjuntos, o operador Union une os itens de dois ou mais conjuntos de dados de entrada gerando um único conjunto unificado. Porém, para aplicar o operador Union não há condição de seleção e, caso alguma seleção seja necessária, deve-se antes aplicar o operador Filter seguido pela aplicação do operador Union.

**Operador Split**

O operador Split representa a separação dos atributos do conjunto de dados fornecido e o direcionamento dos subconjuntos de atributos para fluxos distintos no *workflow*. Assim, a entrada para a operação de Split é unária e a saída é n-ária. Os conjuntos de dados resultantes têm esquemas diferentes contendo quaisquer subconjuntos dos atributos do conjunto original. Pode haver sobreposição de atributos nos resultados, ou seja, um mesmo atributo pode estar contido em dois ou mais subconjuntos do resultado. Os dados resultantes da operação Split podem ser direcionados para outras operações de ETL ou para um repositório.

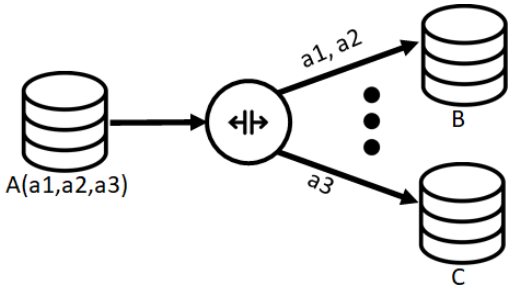



**Figura 20: Cenário de uso do operador Split**

Fonte: Elaborado pela autora

Como exemplo, a Figura 20 representa um possível cenário de uso do operador Split: a entrada é um conjunto de dados A que tem os atributos a1, a2

e a3; o operador Split é aplicado para separar os atributos de A, originando um novo conjunto contendo os atributos a1 e a2 e outro conjunto contendo somente o atributo a3. Supondo que A contém dados de Clientes incluindo informações pessoais, endereço domiciliar, endereço comercial e informações bancárias, o resultado da aplicação Split pode ser, por exemplo, um conjunto B contendo as informações pessoais e os endereços dos Clientes, e o conjunto C contendo as informações bancárias dos Clientes que não serão propagadas para o DW.

### Operador Join

O operador Join é usado para representar a ação de combinar os itens de dados de dois conjuntos fornecidos. Assim, a entrada para o operador Join é binária. Como parâmetro, é obrigatório definir a condição para combinação dos dados (condição de junção) e, opcionalmente, é possível definir uma lista de atributos para a saída (se nenhum atributo for fornecido, então a saída terá todos os atributos dos dois conjuntos de entrada). De forma geral, a saída do Join é unária e corresponde a um novo conjunto de dados contendo os itens do primeiro conjunto que têm correspondência com algum item do segundo conjunto. Opcionalmente, na saída do operador Join, os dados para os quais não foi possível conseguir uma combinação que atendesse à condição de junção fornecida, podem ser direcionados para um fluxo alternativo, ou seja para uma outra tarefa ou para um repositório; assim, nesse exemplo, a saída do Join é binária.

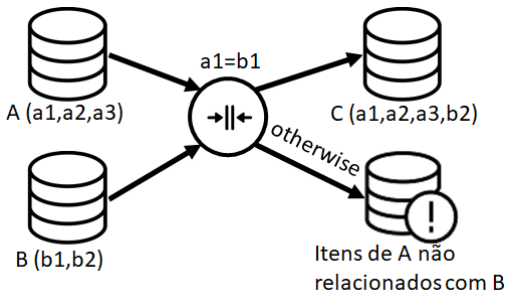



**Figura 21: Cenário de uso do operador Join**

Fonte: Elaborado pela autora

A Figura 21 tem a notação gráfica do operador Join em um cenário onde a entrada é formada pelo conjuntos de dados A, com os atributos a1, a2 e a3, o conjunto de dados B, com os atributos b1 e b2 e a condição de junção a1=b1; como resultado, os itens de dados selecionados, ou seja, aqueles que atendem a condição de junção, são direcionados para o repositório C que, então, contém os atributos a1, a2, a3 e b2. Além disso, nesse exemplo, os itens de dados que não atendem a condição de junção (*otherwise*) são direcionados para um FailDataSet. Supondo que A contém dados de Clientes incluindo o CEP e que B contém dados de Endereço com CEP, e que o critério de junção seja CEP do Cliente = CEP do Endereço, então, o resultado é o conjunto de dados de Clientes incluindo o CEP e as demais informações do Endereço do cliente e, os clientes para os quais não foi possível recuperar o endereço são armazenados em um repositório de falhas.

**Operador Diff**

O operador Diff representa a ação de calcular as diferenças entre os itens de dados de dois conjuntos. Assim, a entrada é binária, composta por dois conjuntos de dados com o mesmo esquema. A saída é unária e consiste no subconjunto de itens do primeiro conjunto que não estão contidos no segundo conjunto, ou seja, as diferenças. O esquema dos dados não é alterado na operação Diff. Considerando que no cálculo de diferenças a ordem das entradas afeta o resultado, o operador Diff requer que cada conjunto de dados de entrada tenha um rótulo de identificação e que, abaixo do símbolo gráfico do operador, haja a indicação da precedência da operação. Além disso, para que a diferença dos dados seja calculada, é necessário que os conjuntos de dados de entrada estejam preparados, ou seja, as operações anteriores aplicadas aos dados devem estar terminadas para que o cálculo das diferenças seja iniciado.

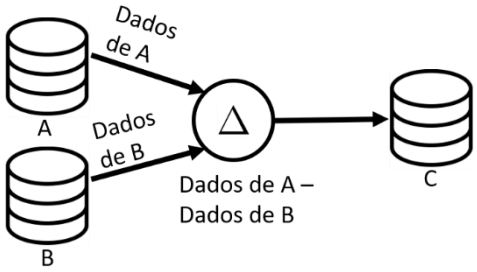



**Figura 22: Cenário de uso do operador Diff**

Fonte: Elaborado pela autora

Como exemplo, a Figura 22 mostra o operador Diff com os conjuntos de dados de entrada A e B, e o resultado que é direcionado para armazenamento em C. Na entrada, os dados apresentam os rótulos Dados de A e Dados de B e a precedência da operação, Dados de A – Dados de B, está indicada. Na prática, o operador Diff é útil para representar a identificação de itens de dados novos, ou seja, aqueles que não haviam sido carregados no DW em um processamento anterior. No domínio de Vendas, por exemplo, supondo que A tem os dados dos clientes que compraram recentemente e que B contém os dados dos clientes que foram carregados no DW no processamento anterior, então, o resultado do Diff é o conjunto dos clientes novos, ou seja, aqueles que ainda não estão no DW.

### Operador Intersect

O operador Intersect indica a recuperação de itens de dados que estão presentes, simultaneamente, nos conjuntos de dados fornecidos. A entrada para a operação Intersect é comumente binária, mas também pode ser n-ária composta por conjuntos de dados que têm, obrigatoriamente, o mesmo esquema. A saída é comumente unária composta por um subconjunto de itens do primeiro conjunto de dados, aqueles itens que estão presentes também nos demais conjuntos, e pode também ser binária quando os itens que não atendem a condição especificada são direcionados para um fluxo alternativo. O esquema dos dados não é modificado. Além disso, fica implícito que para que a intersecção dos dados seja calculada, é necessário que os conjuntos de dados

de entrada estejam preparados, ou seja, as operações anteriores aplicadas aos dados devem estar terminadas para que o cálculo da intersecção seja iniciado.

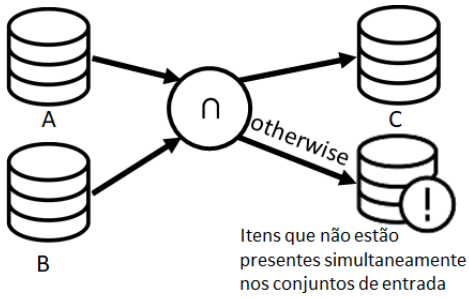



**Figura 23: Operador Intersect com duas saídas**

Fonte: Elaborado pela autora

A figura 23 ilustra o operador Intersect com duas saídas, onde os dados que não atendem a condição são direcionados para um fluxo alternativo.

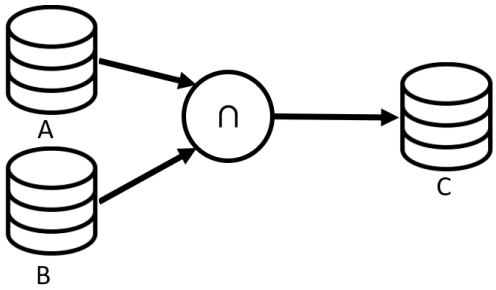



**Figura 24: Cenário de uso do operador Intersect**

Fonte: Elaborado pela autora

A Figura 24 apresenta um cenário de uso do operador Intersect com dois conjuntos de dados de entrada, A e B, e a saída é um conjunto de dados que contém os itens de A que aparecem também em B e é direcionado para o repositório C. Supondo que A contém os dados dos Clientes que compraram no último ano e que B contém os dados dos Clientes que têm cupom de desconto, então o resultado do Intersect é, nesse exemplo, os Clientes que compraram no último ano e que possuem cupom de desconto (os clientes sem cupom de desconto são descartados).

## Operador Sort

O operador Sort realiza a ordenação dos itens de um conjunto de dados, em ordem crescente ou decrescente. A entrada e a saída são ambas unárias. Como parâmetros devem ser fornecida uma lista de atributos do conjunto de entrada, que serão usados na ordenação, e um critério de ordenação (asc ou desc) para cada atributo da lista. O resultado é o mesmo conjunto de dados fornecido, contendo os dados reordenados. O esquema dos dados não é alterado.

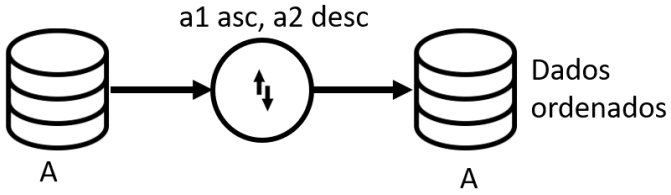



**Figura 25: Cenário de uso do operador Sort**

Fonte: Elaborado pela autora

A Figura 25 apresenta um possível cenário de uso do operador Sort com um conjunto de dados de entrada A, os atributos a1 com critério de ordenação crescente (asc) e a2 com critério de ordenação decrescente (desc), e a saída que é o mesmo conjunto com os dados ordenados. Supondo que A contém os dados de Pedidos, que a1 e a2 são respectivamente Data e Valor do Pedido, então o resultado é o conjunto A com os dados ordenados (i) por Data de forma crescente e (ii) por Valor do Pedido de forma decrescente.

## Operador Update

O operador Update representa a alteração dos valores dos dados de um conjunto. Assim, a entrada e a saída são unárias. Como parâmetro é obrigatório ter uma lista atribuições contendo atributo e valor (exemplo: a←val) e, opcionalmente, uma condição de seleção e, nesse caso, os valores só serão alterados para os itens que atenderem a condição. O esquema dos dados não é alterado na operação Update.

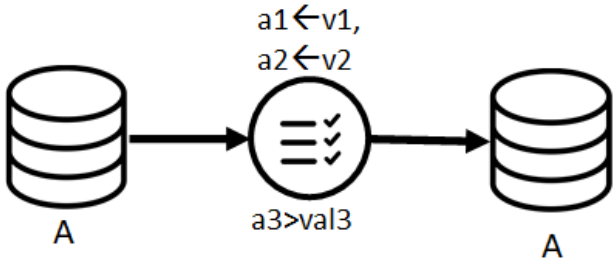



**Figura 26: Cenário de uso do operador Update**

Fonte: Elaborado pela autora

No cenário da Figura 26, os valores dos atributos a1 e a2 são substituídos por v1 e v2, respectivamente, para os itens onde o valor do atributo a3 é maior que v3. Supondo que A contém dados de Pedidos, a1, a2 e a3 são Desconto, Frete e Valor dos pedidos e que v1, v2 e v3 são, 100, 0, 1000, respectivamente, então o resultado contém os dados dos Pedidos que têm Valor maior que 1000, para os quais o Frete foi alterado para 0 e o Desconto foi alterado para 100.

## Operador Copy

A operação Copy representa a geração de uma réplica do conjunto de dados fornecido. Uma possível aplicação ocorre quando há a necessidade de manter um *backup* do conjunto de dados como ponto de recuperação do processo para casos de falhas na execução. A entrada da operação Copy é unária e a saída é binária, ou seja, o próprio conjunto fornecido e uma réplica. A Figura 27 apresenta um possível cenário de uso do operador Copy, com um conjunto de dados A fornecido como entrada e, como resultado, o próprio conjunto A sem modificações e uma réplica.

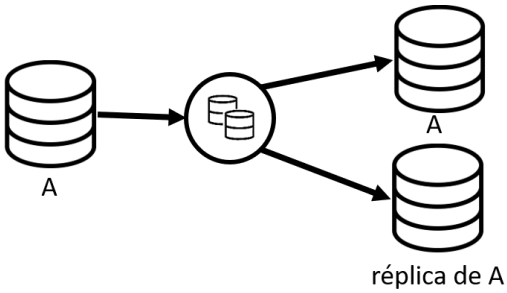



**Figura 27: Cenário de uso do operador Copy**

Fonte: Elaborado pela autora

### 5.2.3 Operadores de agregação

Os operadores de agregação podem ser usados para representar funções que, quando aplicadas a um conjunto de dados, processam os valores e retornam resultados sumarizados. Portanto, a entrada e a saída dos operadores de agregação são, ambas, unárias. Como parâmetro é obrigatório ter uma lista de atributos cujos valores serão analisados e processados e, opcionalmente pode ser fornecida uma condição de seleção e, nesse caso, somente os itens de dados que satisfizerem a condição serão efetivamente considerados na agregação. Para cada operador de agregação há um outro operador correspondente que permite o agrupamento dos dados em grupos distintos e que, portanto, exige um parâmetro adicional que é uma lista de atributos usados para a formação dos grupos.

São propostos 5 operadores de agregação: Sum, Count, Max, Min e Avg e os operadores correspondentes que podem ser usados para o agrupamento dos dados em grupos: SumGroup, CountGroup, MaxGroup, MinGroup e AvgGroup. Esses operadores são apresentados nas próximas seções.

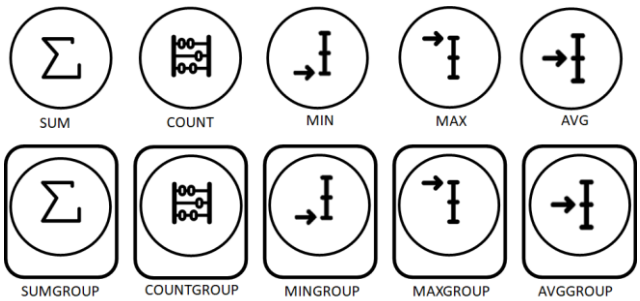



**Figura 28: Notação gráfica dos operadores de agregação**

Fonte: Elaborada pela autora

#### Operadores Sum e SumGroup

O operador Sum indica a soma dos valores de atributos específicos em um conjunto de dados. O resultado é um único valor para cada atributo que corresponde à soma dos valores para cada atributo especificado como

parâmetro. Se uma condição de seleção for fornecida, então apenas os itens de dados que satisfizerem a condição serão considerados na soma.

A Figura 29 apresenta um possível cenário de uso do operador Sum, com um conjunto de dados, A, um atributo at1 e uma condição de seleção at2 = val; a saída é a soma dos valores do atributo at1 fornecido para os itens de dados que satisfazem a condição estabelecida. Supondo que A tenha dados sobre Pedidos, at1 e at2 sejam respectivamente Valor do Pedido e Desconto e que val seja 0, então o resultado, armazenado em B, é um único item de dado que corresponde à soma dos valores dos pedidos, considerando, apenas os pedidos onde Desconto é 0.

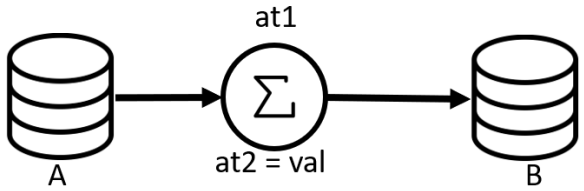



**Figura 29: Cenário de uso do operador Sum.**

Fonte: Elaborada pela autora

Para a representação de grupos de dados há o operador SumGroup que exige como parâmetro uma lista de atributos que são usados para a formação dos grupos.

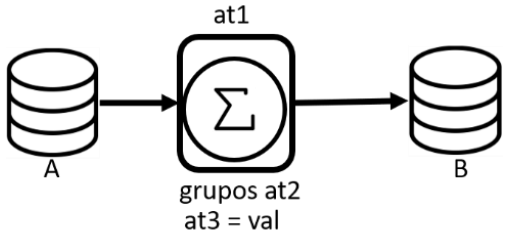



**Figura 30: Cenário de uso do operador SumGroup**

Fonte: Elaborada pela autora

No exemplo da Figura 30, supondo que A tenha dados de Pedidos, que at1, at2 e at3 sejam respectivamente Valor do Pedido, Data do Pedido e Desconto e que val seja 0, então o resultado, armazenado em B é, para cada data, a soma dos valores dos Pedidos que têm Desconto igual a 0.

### Operadores Count e CountGroup

O operador Count é usado para determinar a quantidade de valores distintos para cada atributo especificado em um conjunto de dados. As mesmas explicações realizadas para os operadores Sum e SumGroup se aplicam para os operadores Count e CountGroup, trocando-se a soma pela quantidade no agrupamento.

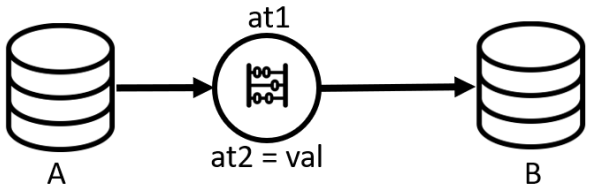



**Figura 31: Cenário de uso do operador Count**

Fonte: Elaborada pela autora

No exemplo apresentado na Figura 31, supondo que A tenha dados de Pedidos e que at1 e at2 sejam o Identificador do Pedido e o Desconto, e que val seja 0, então o resultado da operação, armazenado em B, é a quantidade de pedidos registrados, considerando somente aqueles Pedidos que têm Desconto igual a 0.

Para a representação de agrupamentos há o operador CountGroup que exige como parâmetro uma lista de atributos que são usados para a formação dos grupos.

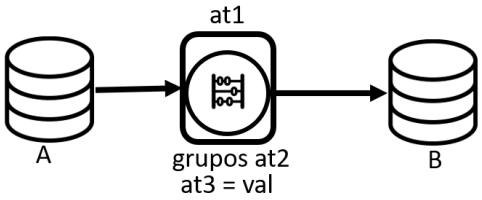



**Figura 32: Cenário de uso do operador CountGroup**

Fonte: Elaborada pela autora

Assim, na Figura 32, supondo que A contém dados de Pedidos, que at1, at2 e at3 são, respectivamente, o Identificador do Pedido, a Data do Pedido e o Desconto, e que val é 0, então o resultado da operação, armazenado em B, é

um valor para cada Data contendo a quantidade de Pedidos com Desconto igual a 0.

### Operadores Min e MinGroup

O operador Min é usado para determinar o valor mínimo para cada atributo especificado em um conjunto de dados. As mesmas explicações realizadas para os operadores Sum e SumGroup se aplicam para os operadores Min e MinGroup, trocando-se a soma pelo valor mínimo no agrupamento.

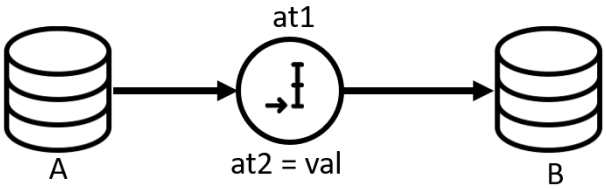



**Figura 33: Cenário de uso do operador Min**

Fonte: Elaborada pela autora

Considerando o exemplo da Figura 33 e supondo que A tenha informações de Pedidos e que at1 e at2 sejam, respectivamente, Valor do Pedido e Desconto e que val seja 0, então B tem um único valor que é o menor valor de Pedido considerando apenas os Pedidos onde Desconto é 0.

Para a representação de agrupamentos há o operador MinGroup que exige uma lista de atributos que são usados para a formação dos grupos.

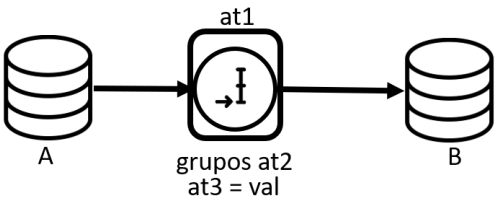



**Figura 34: Cenário de uso do operador MinGroup**

Fonte: Elaborada pela autora

No exemplo da Figura 34, supondo que A tenha informações de Pedidos, que at1, at2 e at3 sejam, respectivamente, Valor do Pedido, Data do Pedido e

Desconto, e que val seja 0, então o resultado da operação, armazenado em B, é um valor para cada Data contendo o menor valor entre os Pedidos com Desconto igual a 0.

### Operadores Max e MaxGroup

O operador Max é usado para determinar o valor máximo de um atributo em um conjunto de dados. As mesmas explicações realizadas para os operadores Sum e SumGroup se aplicam para os operadores Max e MaxGroup, trocando-se a soma pelo valor máximo no agrupamento.

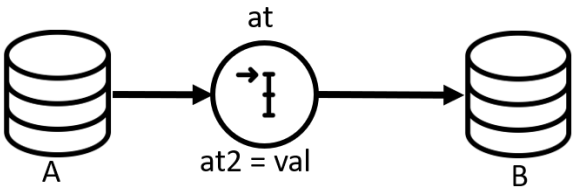



**Figura 35: Cenário de uso do operador Max**

Fonte: Elaborada pela autora

Considerando o exemplo retratado na Figura 35 e supondo que A tenha informações de Pedidos e que at1 e at2 sejam, respectivamente, Valor do Pedido e Desconto, então B tem um único valor que é o maior valor de Pedido onde Desconto é 0.

Para a representação de agrupamentos há o operador MaxGroup que exige uma lista de atributos que são usados para a formação de grupos.

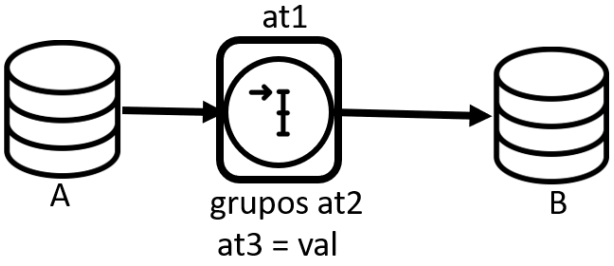



**Figura 36: Cenário de uso do operador MaxGroup**

Fonte: Elaborada pela autora

No exemplo da Figura 36, supondo que A tenha informações de Pedidos, que at1, at2 e at3 sejam, respectivamente, Valor do Pedido, Data do Pedido e Desconto, então o resultado da operação, armazenado em B, é um valor para cada Data contendo o maior valor entre os Pedidos com Desconto igual a 0.

### Operadores Avg e AvgGroup

O operador Avg é usado para determinar o valor médio para cada atributo especificado em um conjunto de dados. As mesmas explicações realizadas para os operadores Sum e SumGroup se aplicam para os operadores Avg e AvgGroup, trocando-se a soma pelo valor médio no agrupamento.

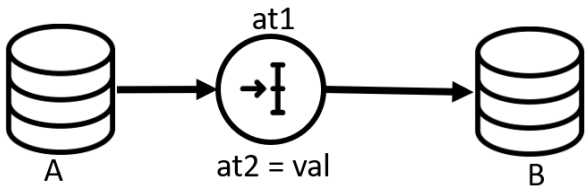



**Figura 37: Cenário de uso do operador Avg**

Fonte: Elaborada pela autora

Considerando o exemplo na Figura 37, supondo que A tenha informações de Pedidos e que at1 e at2 sejam, respectivamente, Valor do Pedido e Desconto, e val é 0, então B tem um único valor que é o valor médio dos pedidos onde Desconto é 0.

Para a representação de agrupamentos há o operador AvgGroup que exige como parâmetro uma lista de atributos que são usados para a formação dos grupos.

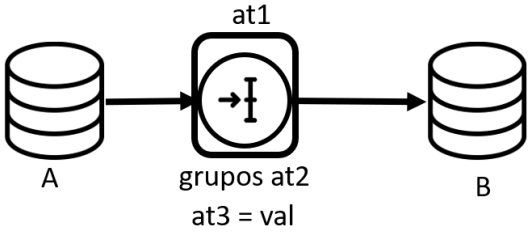



**Figura 38: Cenário de uso do operador AvgGroup**

Fonte: Elaborada pela autora

No exemplo da Figura 38, supondo que A tem informações de Pedidos, que at1, at2 e at3 sejam, respectivamente, Valor do Pedido, Data do Pedido e Desconto, e que val = 0, então o resultado da operação, armazenado em B, é um valor para cada Data contendo o valor médio dos Pedidos onde Desconto é igual a 0.

### 5.2.4 Operadores de inicialização

Os operadores de inicialização de dados servem para representar a inicialização de atributos com valores específicos. São propostos quatro operadores de inicialização para as operações mais comuns em processos de ETL: SetNullAsDefault, SetDefaultValue, SurrogateKey e Sequence. Esses operadores têm entrada e saída, ambas, unárias. A Figura 39 traz a notação gráfica dos operadores de inicialização e a descrição desses operadores é apresentada nas próximas seções.

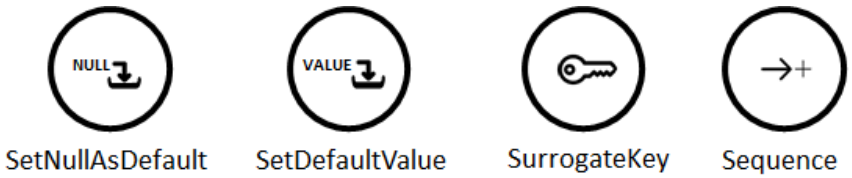



**Figura 39: Notação gráfica dos operadores de inicialização de dados**

Fonte: Elaborada pela autora

#### Operador SetNullAsDefault

O operador SetNullAsDefault é usado para representar a inicialização de um atributo específico com o valor nulo (null), para todos os itens de um conjunto de dados. Como parâmetros, é obrigatório uma lista de atributos que serão inicializados e, opcionalmente, uma condição de seleção que, quando fornecida, indica que somente os itens de dados que satisfizerem a condição serão inicializados. Quando os atributos fornecidos não existirem então eles são criados e inicializados.

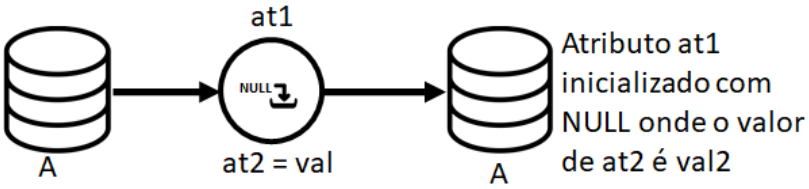



**Figura 40: Cenário de uso do operador SetNullAsDefault**

Fonte: Elaborada pela autora

No exemplo da Figura 40, supondo que A tem informações de Pedidos, que at1 e at2 são, respectivamente, Frete e Desconto e, que val é 0, então o resultado é o mesmo conjunto de dados onde o valor de Frete foi atualizado com nulo para os pedidos com Desconto igual a 0.

### Operador SetDefaultValue

O operador SetDefaultValue é usado para representar a inicialização de atributos com valores específicos. Como parâmetro, é obrigatório ter uma lista de atribuições e, opcionalmente, uma condição de seleção que indica que somente os itens de dados que satisfizerem a condição serão inicializados. Quando os atributos fornecidos nas atribuições não existirem, então eles são criados e inicializados.

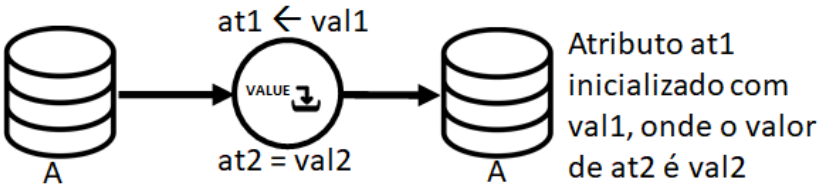



**Figura 41: Cenário de uso do operador SetDefaultValue**

Fonte: Elaborada pela autora

### Operador SurrogateKey

Em um conjunto de dados, um atributo chave é um atributo com valor único e não nulo que pode ser usado para identificação unívoca de cada item do conjunto. Desta forma, garante unicidade no valor dos atributos da chave. Nos processos de ETL, é comum ter a necessidade de criar artificialmente os valores para o atributo chave para garantir a unicidade de cada item de um conjunto de dados. Para essa situação, é proposto o uso do operador SurrogateKey, que tem como parâmetro um atributo chave. O resultado da aplicação do operador SurrogateKey é uma alteração no esquema do conjunto de dados fornecido, que corresponde ao acréscimo do novo atributo chave para o qual é atribuído um valor único gerado automaticamente.

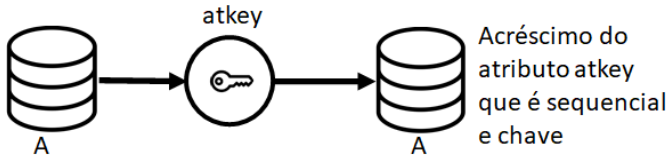



**Figura 42: Cenário de uso do operador SurrogateKey**

Fonte: Elaborada pela autora

A Figura 42 ilustra um cenário de uso do operador SurrogateKey com um conjunto de dados A e o atributo chave atkey e o resultado que é o próprio conjunto de dados A com a inclusão do atributo atkey inicializado com um valor único e sequencial para cada item de dado.

### Operador Sequence

O operador Sequence é um caso particular do operador SurrogateKey que permite representar o acréscimo de um novo atributo não chave a um conjunto de dados, e a atribuição de um valor único e sequencial para esse atributo. Portanto, a diferença da aplicação do operador Sequence com relação ao operador SurrogateKey é a indicação de que o novo atributo não é uma chave, mas um atributo simples, cujo valor foi gerado de forma artificial com a garantia de ter valores sequenciais únicos, ou seja, sem repetição. Assim, como parâmetro deve ser fornecida uma atribuição formada por um atributo e um valor inicial; o resultado é uma alteração no esquema do conjunto de dados fornecido

com o acréscimo do novo atributo inicializado com um valor único gerado automaticamente a partir do valor inicial determinado.

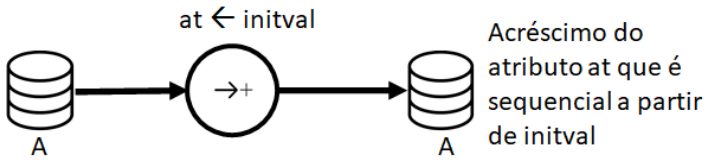



**Figura 43: Cenário de uso do operador Sequence**

Fonte: Elaborada pela autora

A Figura 43 ilustra um cenário de uso do operador Sequence com um conjunto de dados A, o atributo at e o valor inicial initval. O resultado é o conjunto de dados A com a inclusão do atributo at que foi inicializado com um valor único, gerado automaticamente a partir de initval. O operador Sequence é especialmente útil para carregar dados na dimensão Tempo (datas ou horários sequenciais) de um DW.

### 5.2.5 Operadores de fluxo

Os operadores de fluxo de dados permitem representar alterações no fluxo dos dados no *workflow* de ETL, sem impactar esses dados, ou seja, sem alterar o conjunto de dados ou o esquema dos dados. Os operadores propostos nessa categoria são: Fork, Junction, Syncronize, Delay e Fail. A Figura 44 ilustra a notação gráfica proposta para os operadores de fluxo e a descrição desses operadores é apresentada a seguir.

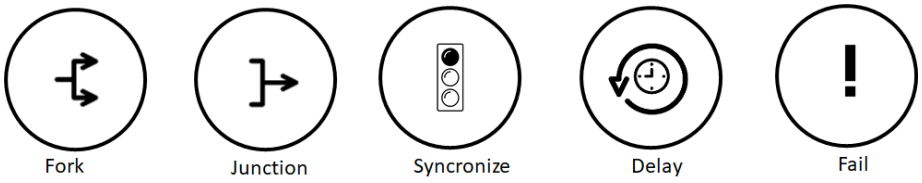



**Figura 44: Notação gráfica dos operadores de fluxo**

Fonte: Elaborada pela autora

**Operador Fork**

O operador Fork é usado para representar uma situação em que um conjunto de dados, vindo de um repositório ou resultante de alguma tarefa, é direcionado para (i) duas ou mais tarefas que são executadas de forma concorrente ou (ii) para repositórios e, também, para tarefas do *workflow*. O operador Fork não impacta os dados que trafegam no *workflow*, apenas altera a sequência linear de execução das tarefas, ou seja, o Fork indica que as tarefas que o sucedem podem ser executadas de forma concorrente. A entrada para o operador Fork é unária e a saída é n-ária (pelo menos binária) e não há parâmetros.

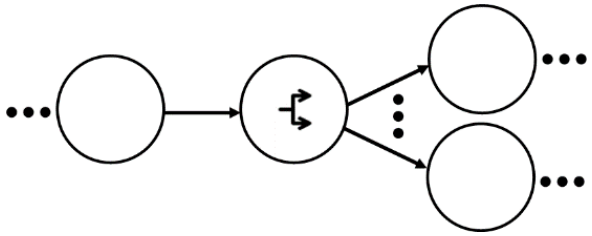


**Figura 45: Cenário de uso do operador Fork**

Fonte: Elaborada pela autora

A Figura 45 ilustra um cenário com o uso do operador Fork: os dados resultantes de transformação ou limpeza entram no operador Fork que os direciona de forma concorrente para outras tarefas do *workflow*. Como exemplo, o operador Fork pode ser usado para que um conjunto de dados seja entregue para diferentes agregações para, posteriormente, serem carregados para *data marts*.

### Operador Junction

O operador Junction representa o ponto de encontro de tarefas que foram realizadas de forma concorrente, ou seja, o ponto de encontro de subfluxos. Assim, a entrada para o operador Junction são os dados resultantes da aplicação de transformações, limpezas ou agregações, e o resultado é o direcionamento dos dados para uma outra tarefa ou para um repositório. Os dados que trafegam no *workflow* não são afetados pelo operador Junction. A entrada para o operador Junction é n-ária (pelo menos binária), a saída é unária e não há parâmetros.

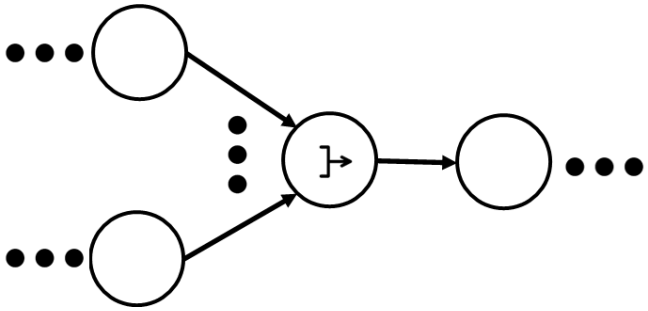


**Figura 46: Cenário de uso do operador Junction**

Fonte: Elaborada pela autora

A Figura 46 ilustra um cenário com o uso do operador Junction: dados resultantes de tarefas concorrentes chegam ao operador Junction e seguem então para uma nova tarefa do *workflow*. Uma situação para o uso do Junction é quando operações de agregação são aplicadas a um conjunto de dados e, ao final dessas operações, os dados resultantes são direcionados para um repositório.

### Operador Syncronize

O operador Syncronize serve para representar um ponto de sincronização onde duas ou mais operações, que estão executando de forma concorrente, aguardam até que estejam finalizadas; os dados resultantes são, então, direcionados para um único fluxo. Em outras palavras, o operador Syncronize indica a existência de uma dependência início-fim entre as operações envolvidas e que, portanto, a operação que sucede a sincronização deve ser iniciada após o término (parcial ou total) das operações concorrentes que a antecedem,

atendendo a alguma condição específica para cada fluxo de entrada. Assim, a entrada para o operador Syncronize é n-ária (pelo menos binária) e a saída é unária. Como parâmetros, o operador Syncronize é obrigatório definir uma condição para cada entrada, indicando quando o fluxo estará apto para continuar e essas condições podem opcionalmente, ser expressas em linguagem algorítmica, como exemplo, “quando a operação de atualização estiver terminada”. Desta forma, o operador Syncronize permite indicar que o fluxo resultante seja iniciado somente quando as condições de todos os fluxos de entrada forem cumpridas, ou seja, ocorre o processamento de um AND lógico entre as condições estabelecidas para cada fluxo de entrada.

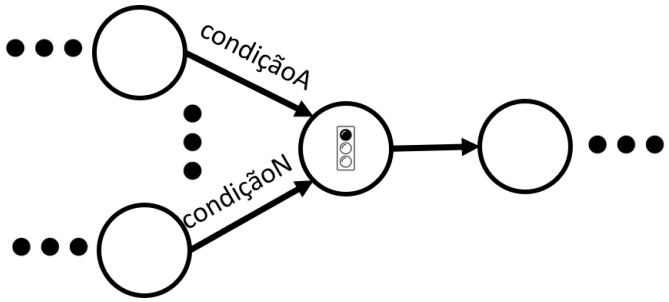



**Figura 47: Cenário de uso do operador Syncronize**

Fonte: Elaborada pela autora

A Figura 47 representa a aplicação do operador Syncronize com vários fluxos de entrada onde cada um tem uma condição, e o fluxo resultante que é iniciado quando as condições estiverem atendidas. Como exemplo, a sincronização é necessária para garantir que uma operação que envolva leitura de dados seja iniciada quando as operações de atualização e de transformação dos dados que a antecedem tenham terminado; nessa situação uma condição é “operação de atualização terminada”, outra condição é “operação de transformação terminada” e ambas têm que ser atendidas para que a operação de leitura seja iniciada.

**Operador Delay**

O operador Delay é o temporizador usado para representar situações em que os dados enviados por um ou mais fluxos concorrentes são analisados em intervalos de tempo pré-definidos ou em horários pré-definidos antes de prosseguir. Por exemplo, os dados de um fluxo podem ser processados a cada hora a fim de prosseguir com a contabilização do maior valor. Adicionalmente, o operador Delay permite representar uma condição para cada fluxo de entrada, de forma que o fluxo resultante ocorra somente quando as condições de entrada forem satisfeitas, ou seja, ocorre o processamento de um AND lógico entre as condições dos fluxos de entrada. Opcionalmente, a condição pode ser expressa em linguagem algorítmica, como exemplo, “quando a operação de atualização estiver terminada”. Caso os intervalos de tempo (ou os horários pré-definidos) sejam diferentes, é necessário aplicar mais de um operador Delay, cada qual com o seu intervalo específico (ou horário pré-definido). Como exemplo, um fluxo de entrada pode ser processado a cada 10 minutos e deve satisfazer a condição de ter pelo menos 10000 itens de dados. Outro exemplo consiste em um fluxo de entrada que deve ser processado sempre às 6h da manhã e deve satisfazer a condição de ter pelo menos 500 itens de dados com atributo X com valor maior que R$ 30000,00.

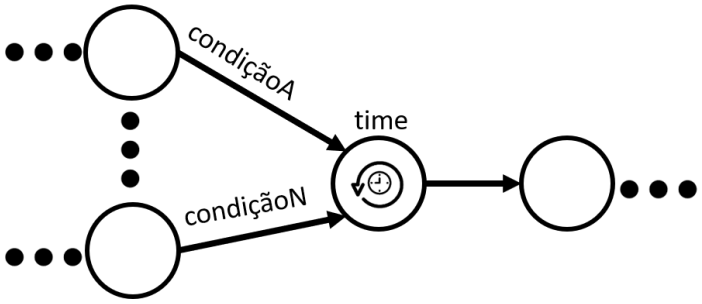



**Figura 48: Cenário de uso do operador Delay**

Fonte: Elaborada pela autora

A Figura 48 ilustra um exemplo de aplicação do operador Delay com várias entradas, uma condição para cada entrada e um intervalo, time; as condições são analisadas conjuntamente quando o tempo é atingido, e o fluxo resultante será iniciado se as condições forem atendidas.

### Operador Fail

O operador Fail pode ser usado para representar um fluxo alternativo em um *workflow* de ETL. A entrada para o operador Fail é unária e é sempre um operador de manipulação de dados, e a saída unária deve ser direcionada para alguma outra operação ou para um repositório que pode ser, por exemplo, um log de erros.

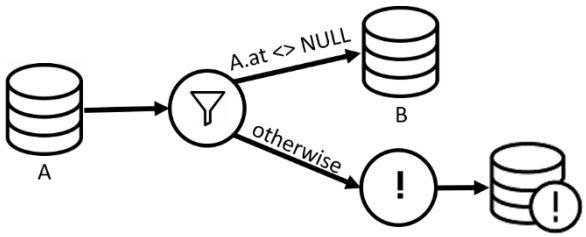



**Figura 49: Cenário de uso do operador Fail**

Fonte: Elaborada pela autora

A Figura 49 ilustra um possível cenário para uso do operador Fail onde o operador Filtro é aplicado para selecionar os dados de A que têm o atributo at preenchido; os dados que não atendem à condição especificada (*otherwise*) são direcionados para o operador Fail e, a partir daí, para um repositório para posterior tratamento.

## 5.2.6 Operadores especiais

Os operadores especiais envolvem especificidades e complementam as funcionalidades dos demais operadores. São dois operadores especiais: Function e Subflow, ilustrados na Figura 50.


**Figura 50: Notação gráfica dos operadores especiais Function e Subflow**

Fonte: Elaborada pela autora

### Operador Function

O operador Function é usado para representar operações ou atividades do de ETL que envolvem especificidades, tais como regras do domínio de negócio, e que não podem ser representadas pelo conjunto de operadores predefinidos no modelo Intuitive. É um recurso importante para garantir a flexibilidade necessária para a construção do *workflow* de ETL, porque permite representar e aplicar operações específicas de um determinado cenário, tais como transformações nos dados de acordo com regras de negócio específicas e muitas vezes complexas.

A entrada do operador Function é n-ária, e pode ser (i) o resultado da aplicação de alguma operação anterior, ou (ii) os dados armazenados em um repositório, ou ainda (iii) o resultado da aplicação de algum outro operador Function. A saída desse operador é tipicamente unária, mas para maior flexibilidade, pode, também ser n-ária.

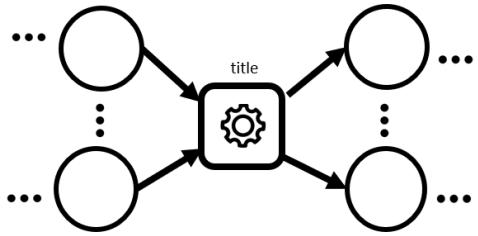



**Figura 51: Uso do operador Function com múltiplas entradas e saídas**

Fonte: Elaborado pela autora

O operador Function tem um título, que indica brevemente qual é a operação implementada e um catálogo descritivo contendo o seu detalhamento: uma descrição curta obrigatória (sentença objetiva) e, quando necessário, uma descrição detalhada (texto longo) e a representação da lógica abstrata na forma de um algoritmo.

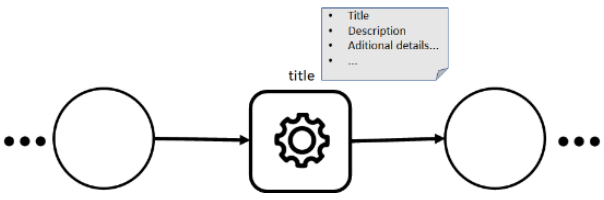



**Figura 52: Cenário de uso do operador Function e catálogo descritivo**

Fonte: Elaborado pela autora

A Figura 52 representa: (i) um cenário onde os dados de entrada passam por uma operação de transformação ou limpeza representada pelo operador Function e o resultado é direcionado para alguma uma outra operação, e (ii) o catálogo descritivo do operador Function.

### Operador Subflow

O operador Subflow permite encapsular partes do *workflow* de ETL que são subfluxos que envolvem conjuntos de operações de ETL. É um recurso importante para melhorar a visualização de *workflows* grandes e complexos. O operador Subflow tem um título que indica o significado das tarefas encapsuladas. A entrada e a saída podem ser ambas n-árias e, quando necessário, pode ter rótulos que são usados para evitar ambiguidades nas entradas e saídas do Subflow, como apresentado na Figura 53, que ilustra o uso desse operador.

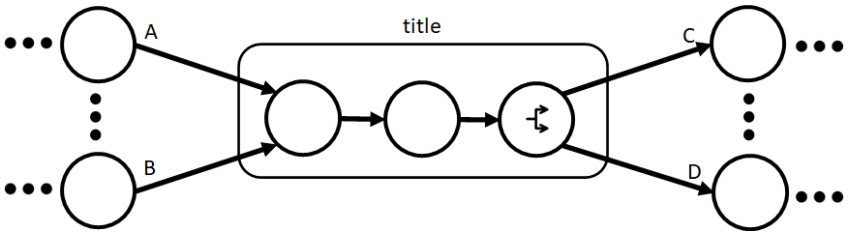
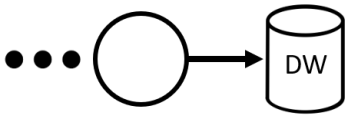
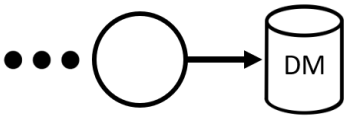
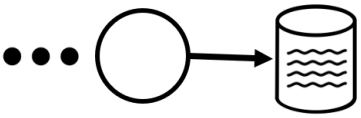
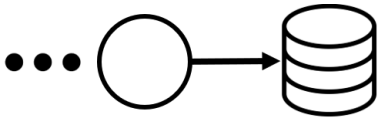
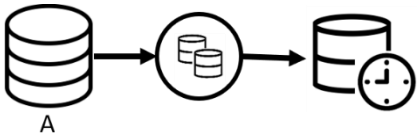
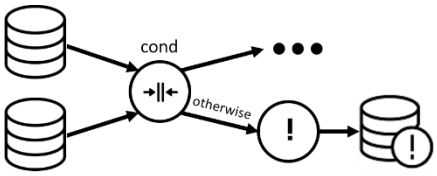
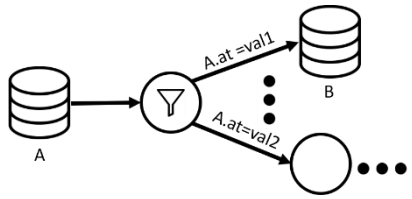
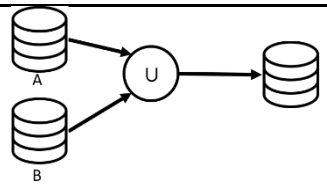
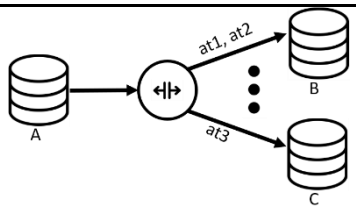
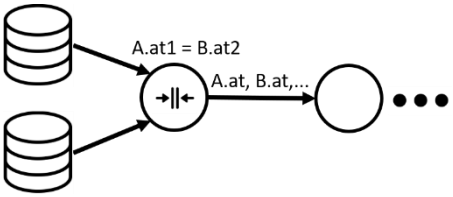
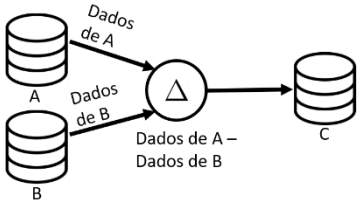
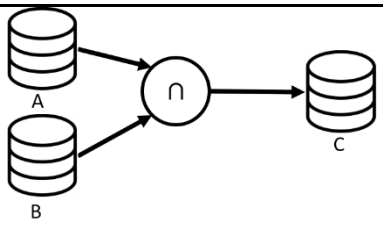
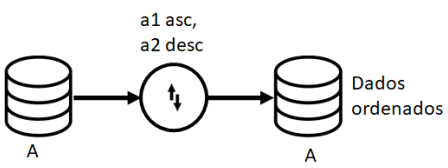


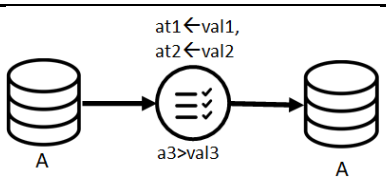
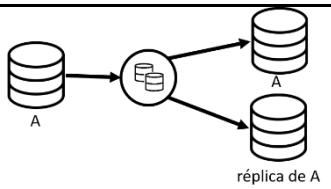
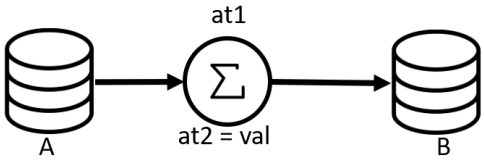
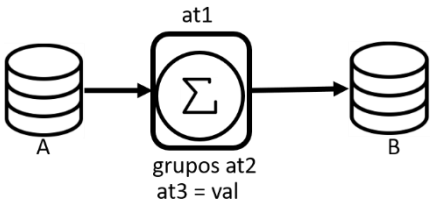
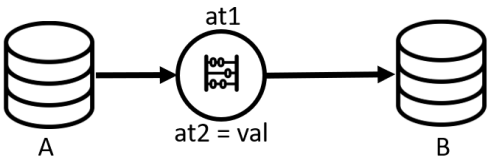
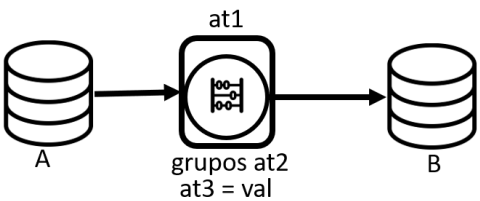

**Figura 53: Uso do operador Subflow com entradas e saídas identificadas por rótulos**

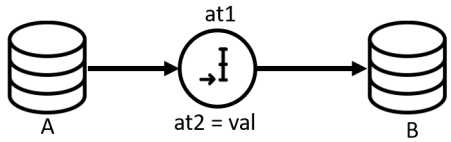
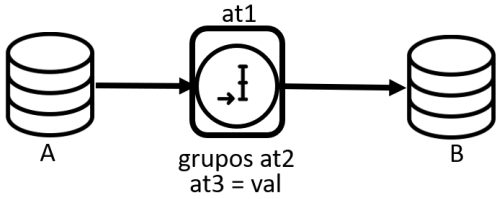
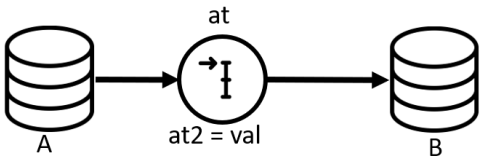
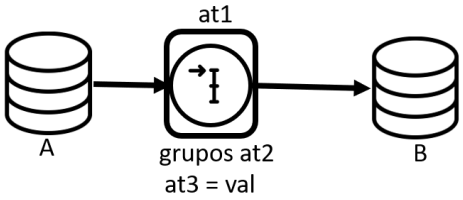
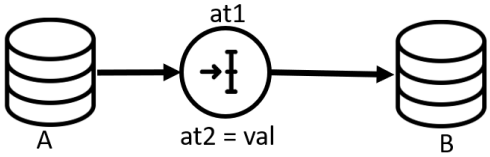
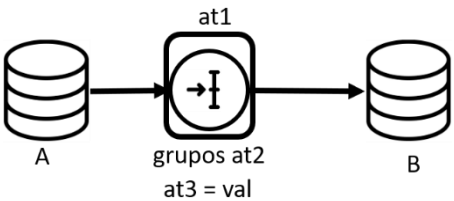
Fonte: Elaborado pela autora

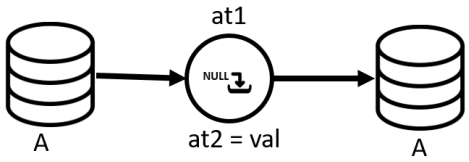
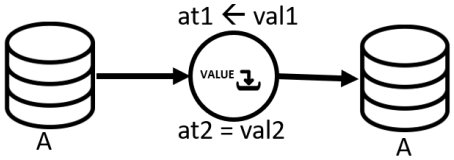
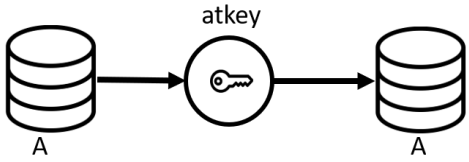
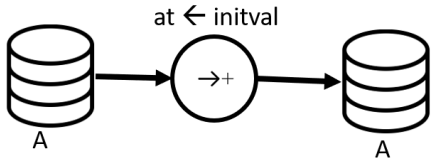
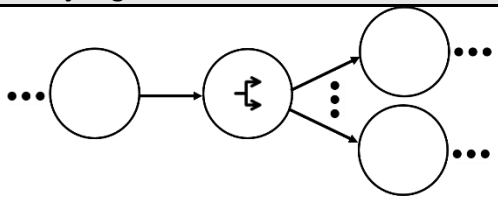
A Tabela 2 apresenta uma síntese dos operadores propostos no modelo Intuitive, contendo o nome, funcionalidade, entradas, parâmetros e saídas.

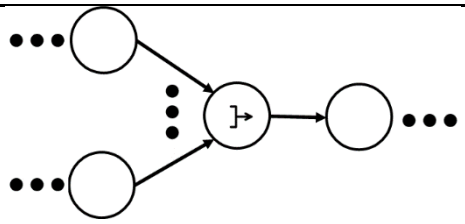
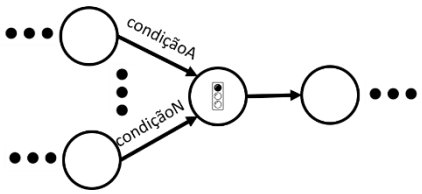
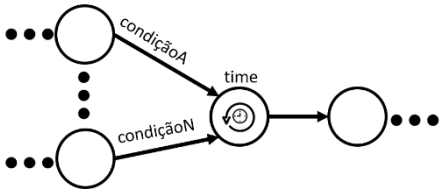
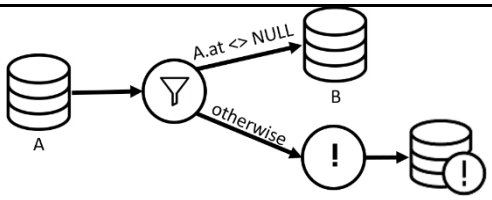
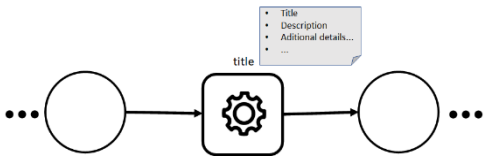
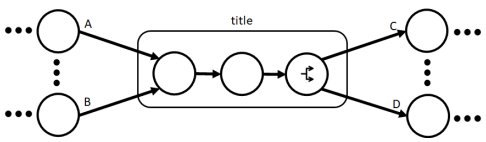
| Operadores de Armazenamento | | | |
|---|---|---|---|
| Operadores | Funcionalidade | Notação gráfica | Entrada, Parâmetros e Saída |
| DataWarehouse | Armazenamento de dados tratados |  | **Entrada**: Unária e não obrigatória – conjunto de dados tratados em tarefas anteriores<br>**Saída**: Unária e não obrigatória – conjunto de dados tratados para agregação, segmentação e consultas. |
| DataMart | Armazenamento de dados segmentados |  | **Entrada**: Unária e não obrigatória – subconjunto de dados do DW.<br>**Saída**: Unária e não obrigatória – dos dados segmentados para compor o DW ou para processamento de consultas OLAP. |
| DataLake | Armazenamento de grande volume de dados brutos |  | **Entrada**: Unária e não obrigatória- dados extraídos de fontes de dados em diversos formatos.<br>**Saída**: Unária e não obrigatória – dados brutos para análise. |
| DataSet | Área de armazenamento tal como repositório, base da dados, tabelas ou arquivos |  | **Entrada**: Unária e não obrigatória – qualquer conjunto de dados resultante da aplicação de tarefas de ETL ou fontes de dados, armazenamento intermediário ou final.<br>**Saída**: Unária e não obrigatória – qualquer conjunto de dados para tarefas de ETL. |
| TempDataSet | Armazenamento temporário de dados |  | **Entrada**: Unária e não obrigatória- qualquer conjunto de dados, armazenado temporariamente para facilitar o processamento.<br>**Saída**: Unária e não obrigatória – dados temporários que apoiam o processamento de uma tarefa de ETL. |
| FailDataSet | Armazenamento de dados rejeitados ou log de erros |  | **Entrada**: Unária– dados resultantes da aplicação de alguma tarefa de ETL que tenham sido rejeitados por não atenderem a alguma condição.<br>**Saída**: Unária e não obrigatória – dados resultantes de um fluxo alternativo para alguma tarefa de ETL. |
| Operadores de Manipulação de Dados | | | |

| Operadores | Funcionalidade | Notação gráfica | Entrada, Parâmetros e Saída |
|---|---|---|---|
| Filter | Seleção de itens de dados |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: Obrigatório - uma condição de seleção para cada saída.<br>**Saída**: n-ária – subconjuntos de itens do conjunto de dados fornecido. |
| Union | União de itens de dados |  | **Entrada**: Binária ou n-ária – conjuntos de dados com o mesmo esquema.<br>**Saída**: Unária – conjunto de dados com todos os itens dos conjuntos fornecidos, sem repetições |
| Split | Separação dos atributos em subconjuntos |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: Obrigatório – uma lista de atributos para cada saída<br>**Saída**: N-ária – subconjuntos dos atributos do conjunto fornecido. |
| Join | Combinação de itens de dados |  | **Entrada**: Binária – quaisquer conjuntos de dados.<br>**Parâmetros**: (i) Obrigatório - condição de junção, (ii) Opcional – lista de atributos para a saída.<br>**Saída**: Unária – conjunto de dados com o esquema alterado. |
| Diff | Diferença entre conjuntos |  | **Entrada**: Binária – dois conjuntos de dados com o mesmo esquema.<br>**Parâmetros**: (i) Obrigatório – um rótulo para cada entrada, (ii) Obrigatório – precedência da operação<br>**Saída**: Unária – itens de dados diferentes entre os conjuntos fornecidos. |
| Intersect | Intersecção entre conjuntos |  | **Entrada**: Binária ou n-ária – conjuntos de dados com o mesmo esquema.<br>**Saída**: Unária ou binária – itens de dados que aparecem em todos os conjuntos fornecidos e, opcionalmente, dados rejeitados |
| Sort | Ordenação dos dados |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: Obrigatório – lista de atributos para ordenação e critério (asc ou desc) para cada atributo da lista. |

| | | | |
|---|---|---|---|
| | | | **Saída**: Unária – o mesmo conjunto de dados fornecido, com os dados ordenados |
| Update | Atualização do valor de algum atributo |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: Obrigatório – lista de atribuições.<br>**Saída**: Unária – mesmo conjunto de dados fornecido, com valores alterados |
| Copy | Criação de uma réplica dos dados |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Saída**: Binária – o próprio conjunto de dados fornecido e uma réplica. |
| Operadores de Agregação | | | |
| Operadores | Funcionalidade | Notação Gráfica | Entrada, Parâmetros e Saída |
| Sum | Soma os valores de atributos determinados |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: (i) Obrigatório – lista de atributos, (ii) condição para a soma.<br>**Saída**: Unária – conjunto de dados com a soma dos valores dos atributos. |
| SumGroup | Soma os valores de atributos determinados para cada grupo |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: (i) Obrigatório – lista de atributos, (ii) lista de atributos para agrupamento.<br>**Saída**: Unária – conjunto de dados com a soma dos valores dos atributos considerando o agrupamento. |
| Count | Quantifica os diferentes valores dos atributos determinados |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: (i) Obrigatório – lista de atributos para quantificação, (ii) condição para a contagem.<br>**Saída**: Unária – conjunto de dados com as quantidades de valores dos atributos. |
| CountGroup | Quantifica os diferentes valores dos atributos determinados para cada grupo |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: (i) Obrigatório – lista de atributos para quantificação, (ii) condição para a contagem , (iii) Obrigatório - lista de atributos para agrupamento.<br>**Saída**: Unária – conjunto de dados com as quantidades de valores dos atributos considerando o agrupamento. |

| | | | |
|---|---|---|---|
| Min | Determinação do menor valor dos atributos determinados |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: (i) Obrigatório – lista de atributos para cálculo, (ii) condição para o cálculo.<br>**Saída**: Unária – conjunto de dados com o menor valor de cada atributo. |
| MinGroup | Determinação do menor valor dos atributos determinados para cada grupo |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: (i) Obrigatório – lista de atributos para cálculo, (ii) condição para o cálculo, (iii) Obrigatório- lista de atributos para agrupamento.<br>**Saída**: Unária – conjunto de dados com o menor valor de cada atributo determinado considerando o agrupamento. |
| Max | Determinação do maior valor dos atributos determinados |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: (i) Obrigatório – lista de atributos para cálculo, (ii) condição para o cálculo.<br>**Saída**: Unária – conjunto de dados com o maior valor de cada atributo. |
| MaxGroup | Determinação do maior valor dos atributos determinados para cada grupo |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: (i) Obrigatório – lista de atributos para cálculo, (ii) condição para o cálculo, (iii) Obrigatório - lista de atributos para agrupamento.<br>**Saída**: Unária – conjunto de dados com o maior valor de cada atributo determinado considerando o agrupamento. |
| Avg | Determinação da média dos valores dos atributos determinados |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: (i) Obrigatório – lista de atributos para cálculo, (ii) condição para o cálculo.<br>**Saída**: Unária – conjunto de dados com o valor médio de cada atributo. |
| AvgGroup | Determinação da média dos valores dos atributos determinados para cada grupo |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: (i) Obrigatório – lista de atributos para cálculo, (ii) condição para o cálculo, (iii) Obrigatório – lista de atributos para agrupamento.<br>**Saída**: Unária – conjunto de dados com o valor médio de cada atributo |

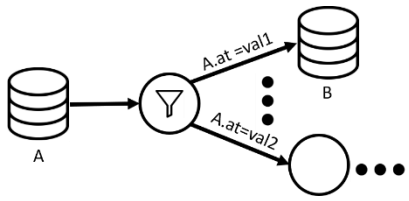
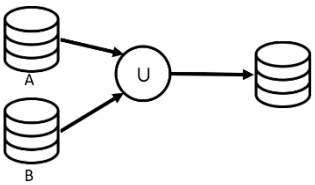
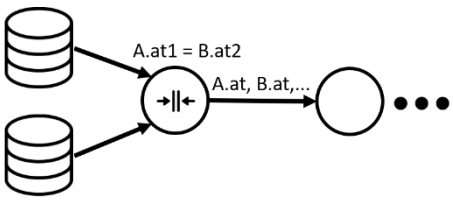
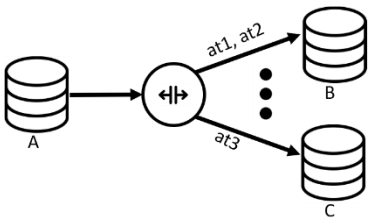
| | | | |
|---|---|---|---|
| | | | determinado considerando o agrupamento. |
| Operadores de Inicialização | | | |
| Operadores | Funcionalidade | Notação gráfica | Entrada, Parâmetros e Saída |
| SetNullAsDefault | Inicialização de atributos com valor nulo |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: (i) Obrigatório – atributo para inicialização, (ii) Opcional – uma condição de seleção de itens para inicialização.<br>**Saída**: Unária – o mesmo conjunto de dados com o atributo inicializado |
| SetDefaultValue | Inicialização de atributos com valores específicos |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: (i) Obrigatório – lista de atribuições, (ii) Opcional – uma condição de seleção de itens para inicialização.<br>**Saída**: Unária – o mesmo conjunto de dados com os atributos inicializados. |
| SurrogateKey | Inicialização de chave artificial |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: Obrigatório – atributo chave.<br>**Saída**: Unária – o mesmo conjunto de dados com a adição do atributo chave. |
| Sequence | Inicialização de atributo com valor sequencial |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Parâmetros**: Obrigatório – uma atribuição inicial.<br>**Saída**: Unária – o mesmo conjunto de dados com o atributo populado com um valor sequencial. |
| Operadores de Fluxo | | | |
| Operadores | Funcionalidade | Notação gráfica | Entrada, Parâmetros e Saída |
| Fork | Submete um conjunto de dados para tarefas paralelas |  | **Entrada**: Unária – qualquer conjunto de dados.<br>**Saída**: Binária ou n-ária – tarefas ou armazenamentos executados em paralelo. |

| | | | |
|---|---|---|---|
| Junction | Encontro de tarefas que executam em paralelo |  | **Entrada**: Binária ou n-ária – conjuntos de dados resultantes de tarefas de ETL que executaram em paralelo.<br>**Saída**: Unária – um único conjunto de dados. |
| Syncronize | Sincronização por meio da espera para a execução de uma atividade em períodos específicos (a cada hora) ou horários específicos (em certa hora). |  | **Entrada**: Binária ou n-ária – tarefas que executarão em paralelo.<br>**Parâmetros**: Obrigatório – uma condição de espera para cada entrada (todas devem ser cumpridas).<br>**Saída**: Unária – um único conjunto de dados |
| Delay | Espera para a execução de uma atividade |  | **Entrada**: Unária – um conjunto de dados qualquer<br>**Parâmetros**: Obrigatório – uma condição de espera (que deve ser cumprida.<br>**Saída**: Unária – o conjunto de dados fornecido |
| Fail | Indicação de um fluxo alternativo |  | **Entrada**: Unária – conjunto de dados resultantes da falha na execução de alguma tarefa.<br>**Saída**: Unária – o mesmo conjunto de dados fornecido |
| Operadores Especiais | | | |
| Operadores | Funcionalidade | Notação gráfica | Entrada, Parâmetros e Saída |
| Function | Função ou rotina que implementa alguma regra de negócio específica |  | **Entrada**: N-ária – conjuntos de dados que serão processados.<br>**Parâmetros**: (i) Obrigatório – título da função, (ii) Opcional – catálogo descritivo – detalhamento da rotina ou função.<br>**Saída**: Comumente unária, mas pode ser n-ária – conjuntos de dados resultantes da aplicação da função ou rotina. |
| Subflow | Encapsulamento de partes do *workflow* do ETL |  | **Entrada**: Comumente unária, mas pode ser n-ária – conjuntos de dados sobre os quais serão aplicadas tarefas de transformação, limpeza ou agregação.<br>**Parâmetros**: (i) Obrigatório – título do subfluxo. (ii) Obrigatório para entradas múltiplas – um rótulo para cada entrada, (iii) Obrigatório para saídas múltiplas – um rótulo para cada saída. |

| | | | |
|---|---|---|---|
| | | | **Saída**: Comumente unária, mas pode ser n-ária – conjuntos de dados resultantes da aplicação de tarefas de transformação, limpeza ou agregação |

**Tabela 2. Síntese da apresentação dos operadores do modelo Intuitive**

Alguns operadores do modelo Intuitive têm funcionalidades inversas ou complementares. A Tabela 3 apresenta a complementariedade existente entre esses operadores.

| Operadores complementares | | | | |
|---|---|---|---|---|
| Operador | | Complemento | | Caracterização |
| Filter |  | Union |  | **Filter** separa os itens de um conjunto de dados fornecido dando origem a dois ou mais subconjuntos.<br>**Union** une os itens de dois ou mais conjuntos de dados gerando um conjunto unificado.<br>O Union não tem parâmetros e então se for necessário fazer uma seleção deve-se antes aplicar o Filter seguido pela aplicação do Union. |
| Join |  | Split |  | **Join** combina os itens de dados de conjuntos que têm esquemas diferentes, formando um novo conjunto com atributos dos conjuntos fornecidos.<br>**Split** separa atributos de um conjunto de dados, gerando subconjuntos desses atributos; exige uma lista de atributos para cada saída. |

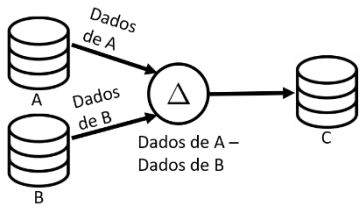
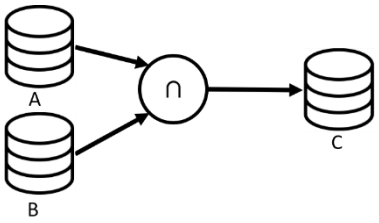
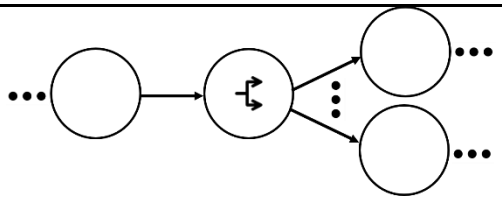
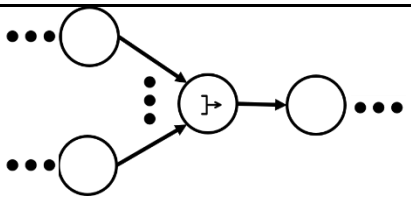
| | | | | |
|---|---|---|---|---|
| Diff |  | Intersect |  | **Diff** calcula e retorna as diferenças entre conjuntos de dados com o mesmo esquema.<br>**Intersect** calcula e retorna os itens que estão presentes nos conjuntos de dados fornecidos e que têm o mesmo esquema. |
| Fork |  | Junction |  | **Fork** submete um conjunto de dados para tarefas paralelas.<br>**Junction** é o encontro de tarefas que executam em paralelo. |

**Tabela 3. Complementariedade entre os operadores do modelo Intuitive**

## 5.3 **Considerações finais**

Nesse capítulo foi apresentada, de forma detalhada, a proposta do modelo Intuitive, o qual tem por objetivo prover suporte à modelagem conceitual de *workflows* de ETL. O modelo Intuitive é baseado em um conjunto de operadores representados por elementos gráficos. Foram propostas seis categorias de operadores: operadores de armazenamento, manipulação de dados, de agregação, de inicialização, de fluxo e de operadores especiais. A validação do modelo Intuitive e os resultados obtidos são descritos no capítulo 6.

# Capítulo 6

# Validação do Modelo Intuitive

*Esse capítulo detalha as atividades realizadas na etapa de validação do modelo Intuitive, envolvendo a seleção de características de qualidade, a seleção de cenários de ETL para a aplicação dos operadores e a elaboração de diagramas. São apresentados também as análises e os experimentos realizados, a coleta dos dados, os resultados obtidos e as conclusões.*

## 6.1 **Considerações iniciais**

Para a validação do Modelo Intuitive, foi estabelecida uma lista de características de qualidade intragramaticais que podem ser aplicadas a modelos conceituais em geral e, em particular, a modelos conceituais de ETL. Essa etapa foi importante para que as características definidas pudessem ser usadas como parâmetros nas atividades de validação do Modelo Intuitive, ajudando na comparação entre os modelos conceituais de ETL encontrados na literatura. Além disso, foi necessário estabelecer um conjunto de cenários de ETL representativos para serem usados nas atividades de validação.

Foram realizadas 4 atividades de validação complementares entre si: (i) uso dos operadores para a modelagem de *workflows* de ETL, (ii) comparação teórica do estado da arte para a modelagem conceitual de *workflows* de ETL com a proposta descrita nesta dissertação de Mestrado, (iii) experimento prático, com

a participação de usuários, para análise comparativa de cenários de ETL representados usando tanto o estado da arte para modelagem conceitual quando o modelo Intuitive, envolvendo características de qualidade pré-estabelecidas e (iv) experimento prático, com a participação de usuários, para verificar a usabilidade do modelo Intuitive, aplicando-o na construção de esquemas conceituais. As atividades realizadas para a validação do modelo Intuitive estão descritas nas próximas seções desse capítulo.

## 6.2 **Seleção dos cenários de ETL**

Um desafio enfrentado na proposição das atividades de validação foi a seleção de cenários de ETL que pudessem ser aplicados. Foram considerados os cenários descritos em SIMITSIS et al. (2009), como parte da proposta de um *benchmarking* de *workflows* de ETL, que são baseados nos padrões TPC-R/TPC-H. Os cenários propostos tratam de informações de Fornecedores de Peças (Figura 54), e representam situações de ETL que ocorrem comumente em ambientes de *data warehousing*. Além disso, em SIMITSIS et al. (2009) foi possível obter as informações necessárias para simular a fase de modelagem conceitual como parte da construção de ETL: a modelagem conceitual ocorre no início da construção do processo de ETL quando decisões técnicas ainda não foram tomadas, tais como ferramentas de automação, decisões arquiteturais, linguagens de programação, ou seja, de forma geral, as informações disponíveis para a modelagem conceitual de processo de ETL são compostas apenas por (i) descrição das fontes de dados, (ii) descrição da estrutura proposta para o DW, (iii) mapeamento dos principais requisitos de negócios que devem ser atendidos pelo ambiente de *data warehousing*.

## 6.3 **Uso do modelo Intuitive para modelagem de cenários de ETL**

A aplicação do modelo Intuitive na modelagem de cenários de ETL foi realizada para verificar se os operadores propostos são suficientes para representar cenários que aparecem com frequência em *workflows* de ETL. Outros objetivos dessa atividade foram melhorar a percepção sobre o atendimento de requisitos relevantes para os esquemas conceituais de *workflows* de ETL, a saber: (i) a simplicidade e a facilidade para a compreensão dos esquemas, visando contribuir para melhorar a comunicação dos projetistas e desenvolvedores com os usuários da área de negócios, (ii) o esforço necessário para elaboração dos esquemas, visando agilizar a fase inicial da construção do processo de ETL, (iii) a facilidade oferecida pelo modelo Intuitive para elaboração de simulações e analisar o impacto de mudanças necessárias para o *workflow*, e (iv) o potencial do modelo Intuitive, como forma de documentação das decisões tomadas durante a construção do processo de ETL em um ambiente de *data warehousing*. A descrição das fontes de dados e do DW, juntamente com a descrição dos cenários de ETL que foram alvo da atividade de validação são apresentados a seguir, bem como o esquema conceitual elaborado para cada cenário.

| | |
|---|---|
| **Data Warehouse** | **PART**(rkey, s_partkey, name, brand, type, size, container, comment)<br>**SUPPLIER**(s_suppkey, name, address, nationkey, phone, acctbal, comment, totalcost)<br>**PARTSUPP**(s_partkey, s_suppkey, availqty, supplycost, comment))<br>**CUSTOMER**(s_custkey, name, address, nationkey, phone, acctbal, mktsegment, comment)<br>**ORDER**(s_orderkey, custkey, orderstatus, totalprice, orderdate, orderprioritu, clerk, shippriority, comment)<br>**LINEITEM**(s_orderkey, partkey, suppkey, linenumber, quantity, extendedprice, discount, tax, returnflag, linestatus, shipdate, commitdate, receiptdate, shipinstruct, shipmode, comment, profit) |
| **Storage House** | **PART**(partkey, name, mfgr, brand, type, size, container, comment)<br>**SUPPLIER**(suppkey, name, address, nationkey, phone, acctbal, comment)<br>**PARTSUPP**(partkey, suppkey, availqty, supplycost, comment) |
| **Sales Point** | **CUSTOMER**(custkey, name, , address, nationkey, phone, acctbal, mktsegment, comment)<br>**ORDER**(orderkey, custkey, orderstatus, totalprice, orderdate, orderprioritu, clerk, shippriority, comment)<br>**LINEITEM**(orderkey, partkey, suppkey, linenumber, quantity, extendedprice, discount, tax, returnflag, linestatus, shipdate, commitdate, receiptdate, shipinstruct, shipmode, comment, profit) |

**Figura 54: Estrutura das fontes de dados e do DW, no benchmarking de ETL**

Fonte: Adaptada de Simitsis et al., 2009

### Cenário 1: *Line*

O cenário *Line* representa um fluxo único e linear que tem uma sequência de operações e, também, áreas de armazenamento de dados, sendo que todas as operações têm exatamente uma entrada e uma saída (entrada e saída unárias). As operações são seleções de dados, transformações e agregações que são aplicadas a um único conjunto de dados de origem para torná-los compatíveis com a estrutura do DW. Para a realização do experimento, o escopo do cenário foi limitado até a carga dos dados no DW e, portanto, não foram consideradas as agregações usadas para popular os *data marts* que aparecem no cenário original, pois essas etapas são parte do processo de customização do DW em *data marts* (construção *top-down*) e, portanto, fora do escopo do processo de ETL. Os seguintes passos são realizados:

- A fonte dos dados é LineItem.D+.
- Nos dados originais, se os atributos "partkey", "orderkey" e "suppkey" estiverem vazios, devem ser iniciados com um valor *default*.
- Os valores de "extendeddprice", "tax", "discount" e "profit", que são valores monetários em Dólar, devem ser convertidos para a moeda Euro.
- O valor de “profit’ deve ser calculado a partir dos valores de "extendeddprice", "tax", "discount".
- Os dados resultantes da aplicação das operações devem ser carregados para DW.D+ (carga total) e, depois em DWH (carga incremental, ou seja, somente atualização dos dados e inclusão de dados novos).

A Figura 55 retrata o esquema conceitual que foi elaborado com o modelo Intuitive para o cenário *Line*.

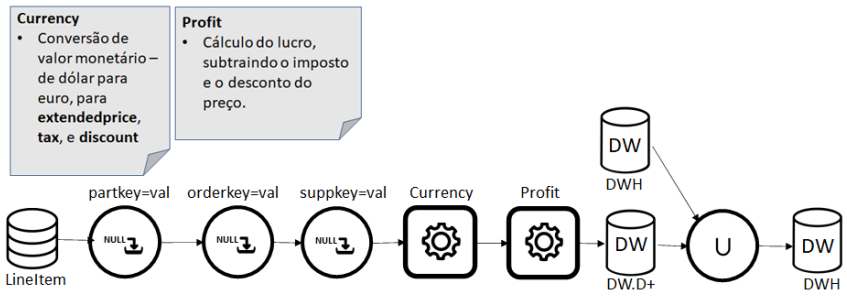



**Figura 55: Esquema conceitual do cenário *Line*, usando o modelo Intuitive**

Fonte: Elaborado pela autora

## Cenário 2: *Wishbone*

O cenário *Wishbone* representa a união de dois fluxos lineares e paralelos, que ocorre quando os dados extraídos de duas fontes são unidos e carregados no DW. A Figura 56 retrata o esquema conceitual que foi elaborado com o modelo Intuitive para o cenário *Wishbone*. No exemplo, os dados de Cliente são extraídos de Customer.New e, depois de unidos aos dados de Customer.Old, são carregados no DW; os seguintes passos são aplicados:

- Seleção dos dados de Customer.New, onde “custkey” não é nulo; os demais dados são considerados inválidos e direcionados para um repositório para posterior tratamento.
- Nos dois fluxos (New e Old) é aplicada a criação da chave artificial “custkey”.
- Nos dois fluxos, os dados de telefone são formatados e padronizados.
- Os dados dos dois fluxos são comparados (New – Old) e as diferenças (clientes novos) são carregados para o repositório temporário C.D+ (carga total) e, a partir daí, os dados são carregados para o repositório Customer (carga incremental).

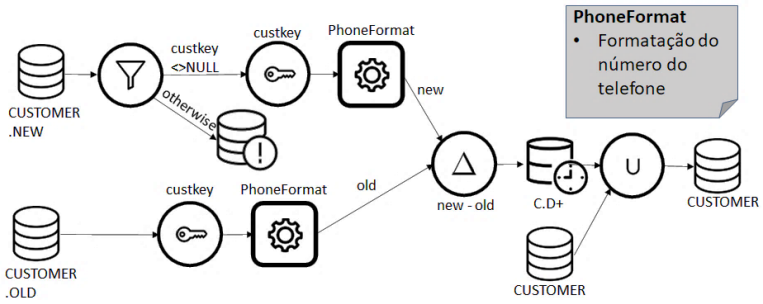



**Figura 56: Esquema conceitual do cenário *Wishbone* com o modelo Intuitive**

Fonte: Elaborado pela autora

## Cenário 3: *Primary Flow*

O cenário *Primary Flow* representa a situação em que os dados originais estão normalizados, sendo necessário desnormalizá-los, ou seja, selecionar as informações correspondentes que estão armazenadas em repositórios auxiliares e juntá-las aos dados originais. Nesse exemplo, a partir da fonte de dados Order, são aplicados os seguintes passos:

- Os atributos “orderstatus”, “custkey”, “orderkey” são usados para sucessivas junções com as informações de L.status, L.cust e L.ord.

- O conjunto de dados resultante das operações de junção são usados para atualização do DW.Order (carga incremental).

A Figura 57 retrata o esquema conceitual que foi elaborado com o modelo Intuitive para o cenário *Primary Flow.*

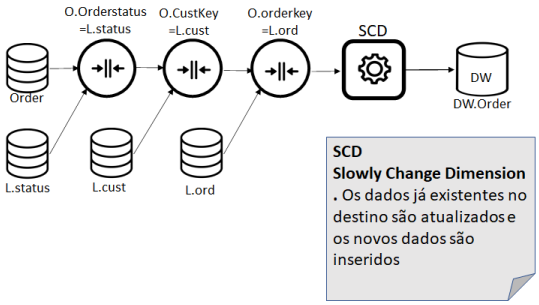



**Figura 57: Esquema conceitual do cenário *Primary Flow* com o modelo Intuitive**

Fonte: Elaborado pela autora

## Cenário 4: *Tree*

O cenário *Tree* representa a união dos dados obtidos de duas ou mais fontes e a aplicação de operações de agregação. Para esse exemplo são considerados 3 fontes de dados (PS1, PS2 e PS3) e o DW PartSupp. Os passos envolvidos são:

- A partir dos dados de entrada, as chaves primárias, “pkey” e “suppkey”, são usadas para comparar com os dados do processamento anterior e seleção de dados novos ou atualizados.
- Para os dados selecionados é aplicada a criação de chave artificial “custkey”.
- Os dados dos 3 fluxos são unidos e ordenados pelo atributo “pkey”.
- Os dados resultantes são carregados para o repositório temporário PS.D (carga total) e, a partir daí, os dados são carregados para o DW, DW.PS (carga incremental).

A Figura 58 retrata o esquema conceitual que foi elaborado com o modelo Intuitive para o cenário *Tree.*

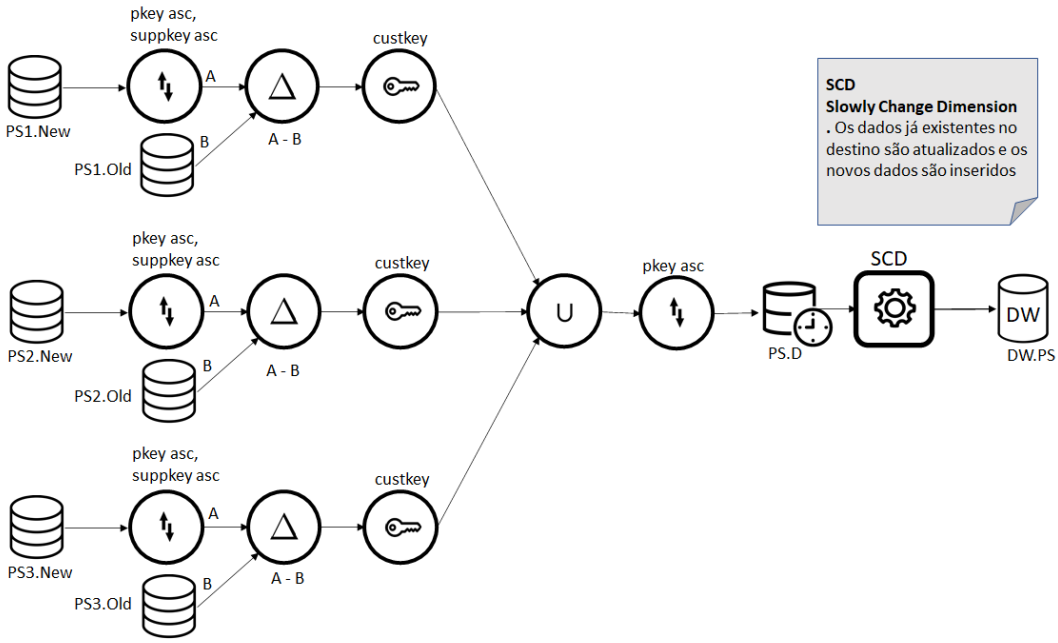



**Figura 58: Esquema conceitual do cenário *Tree* com o modelo Intuitive**

Fonte: Elaborado pela autora

## Cenário 5: *Fork*

O cenário Fork representa a aplicação de agregações aos dados de uma única fonte, e os dados resultantes são propagados para os *data marts*. Nesse exemplo, a fonte é Lineitem e são aplicados os seguintes passos:

- Criação de chaves artificiais para "partkey", "orderkey" e "suppkey".
- Conversão das datas "shipdate" e "receiptdate" para o identificador “dateId” que é único para cada data.
- Cálculo de “profit”, a partir de “extendedprice” com a subtração de “tax” e “discount”.
- Conversão dos valores monetários de Dólar para Euro.
- Armazenamento dos dados resultantes em D+.LI (carga total) e, a partir daí, atualização dos dados no DW.LI (carga incremental).
- Os dados então passam por agregações e são propagados para os *data marts* V015, V06, V07 e V08.

A Figura 59 retrata o esquema conceitual que foi elaborado com o modelo Intuitive para o cenário *Fork*.

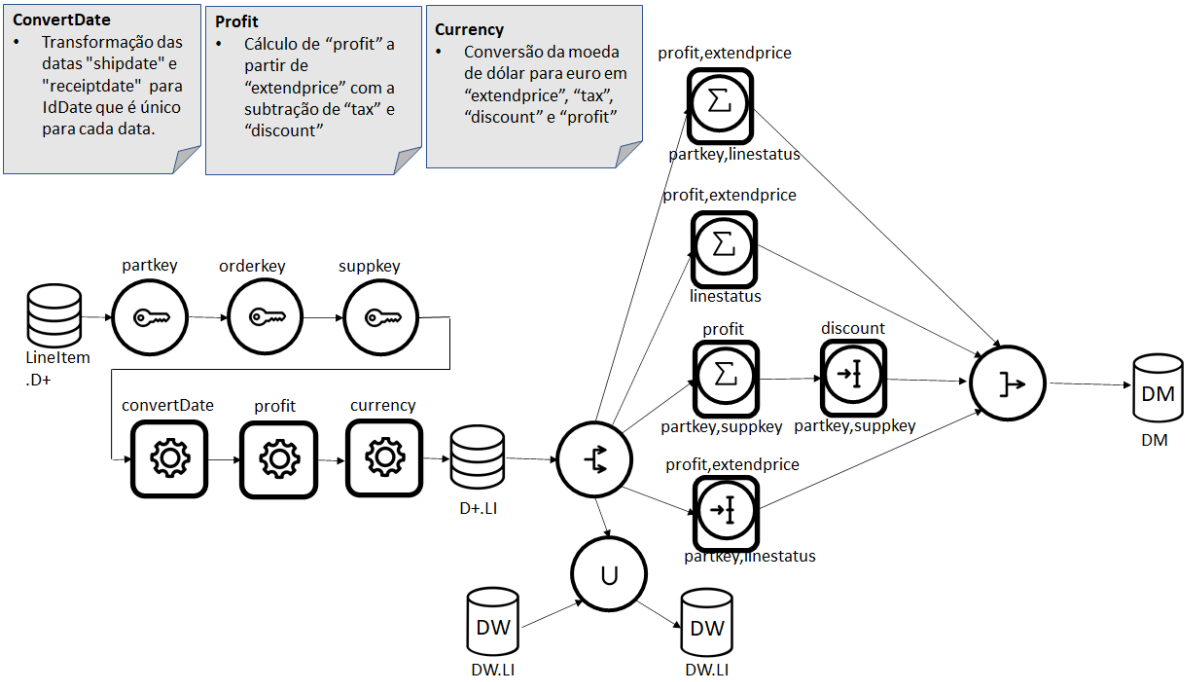



**Figura 59: Esquema conceitual do cenário *Fork* com o modelo Intuitive**

Fonte: Elaborado pela autora

## Conclusões

A partir dos esquemas conceituais obtidos para os cenários anteriormente descritos (Figuras 55 a 59), conclui-se que os operadores do modelo Intuitive foram suficientes para representar os cenários que foram descritos em um *benchmark* de *workflows* de ETL (SIMITSIS et al., 2009). Os esquemas resultantes são simples e de fácil compreensão. É possível fazer simulações de cenários alternativos por exemplo, alterando a precedências das operações ou incluindo outras operações necessárias, e o esquema serve para documentar as deciões tomadas durante a etapa de modelagem. O operador Function permite o tratamento de regras específicas do domínio do negócio, que podem ser reutilizadas, o que garante a possibilidade de adaptação do modelo a novos requisitos específicos decorrentes das regras de negócio. O uso de um catálogo para descrever cada operador Funcion melhora a compreensão do *workflow* de ETL. Alguns operadores propostos não foram necessários para modelar os cenários descritos neste experimento e, portanto, é necessário aplicá-los na construção de variações desses cenários para confirmar a relevância deles.

## 6.4 **Validação teórica**

O objetivo da realização dessa atividade foi avaliar, de forma comparativa, o modelo Intuitive com relação ao estado da arte em modelagem conceitual de processos de ETL que corresponde ao trabalho de Vassiliadis, Simitsis e Skiadopoulos (2002) e, para a comparação, foi usado o cenário de ETL descrito naquele trabalho. Esse cenário é do domínio de Fornecedores de Peças, para o qual já existe o esquema conceitual com o uso do modelo de Grafos, e o esquema lógico, com o uso do modelo de Grafo de Arquitetura. Apesar de proposta ser específica para a modelagem no nível conceitual, foi necessário adicionar o esquema lógico para permitir maior clareza na compreensão do *workflow* de ETL produzido pelo estado da arte. A estrutura das fontes de dados e do destino é apresentada na Figura 60, que foi extraída de Vassiliadis, Simitsis e Skiadopoulos (2002).

| FONTE | NOME | ESQUEMA |
|---|---|---|
| S1 | S1.PARTSUPP | PKEY, DATE, QTY, COST |
| S2 | S2.PARTSUPP | PKEY, QTY, COST |
| DSA | DS.PS_NEW1 | PKEY, DATE, QTY, COST |
| | DS.PS_OLD1 | PKEY, DATE, QTY, COST |
| | DS.PS1 | PKEY, DATE, QTY, COST |
| | DS.PS_NEW2 | PKEY, QTY, COST |
| | DS.PS_OLD2 | PKEY, QTY, COST |
| | DS.PS2 | PKEY, QTY, COST |
| DW | DW.PARTDUPP | PKEY, SUPPKEY, DATE, QTY, COST |
| | LOOKUP_PS | PKEY, SOURCE, SKEY |
| | V1 | PKEY, DAY, MIN_COST |
| | V2 | PKEY, MONTH, AVG_COST |
| | TIME | DAY, MONTH, YEAR |

**Figura 60: Estruturas da fontes de dados e do DW do exemplo usado na etapa de validação teórica**

Fonte: Adaptada de Vassiliadis, Simitsis e Skiadopoulos (2002)

O esquema conceitual com Grafo e o esquema lógico com Grafo de Arquitetura para o exemplo usado nessa etapa de validação foram extraídos de Vassiliadis, Simitsis e Skiadopoulos (2002) e estão representados nas Figuras 61 e 62.

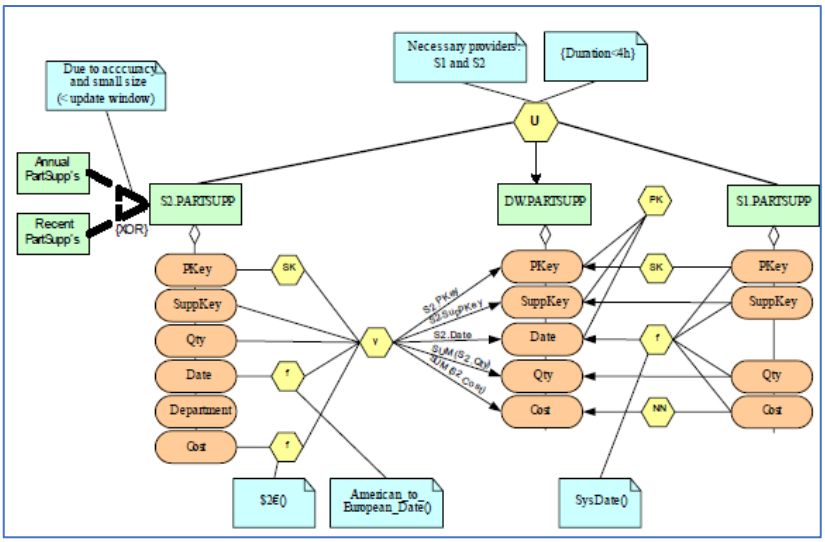



**Figura 61: Esquema conceitual com uso do modelo de Grafo para o cenário de ETL usado como referência para a validação teórica**

Fonte: Extraída de Vassiliadis, Simitsis, Skiadopoulos, 2002

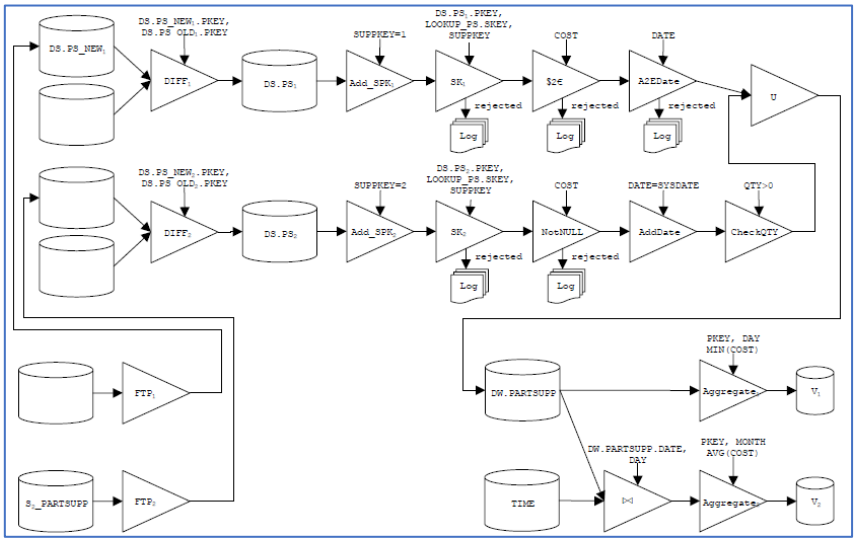



**Figura 62: Esquema lógico com uso do modelo Grafo de Arquitetura para o cenário de ETL usado como referência para a validação teórica**

Fonte: Extraída de Vassiliadis, Simitsis, Skiadopoulos, 2002

Assim, para o cenário do exemplo selecionado, foi elaborado o esquema conceitual com o modelo Intuitive que é apresentado na Figura 63.

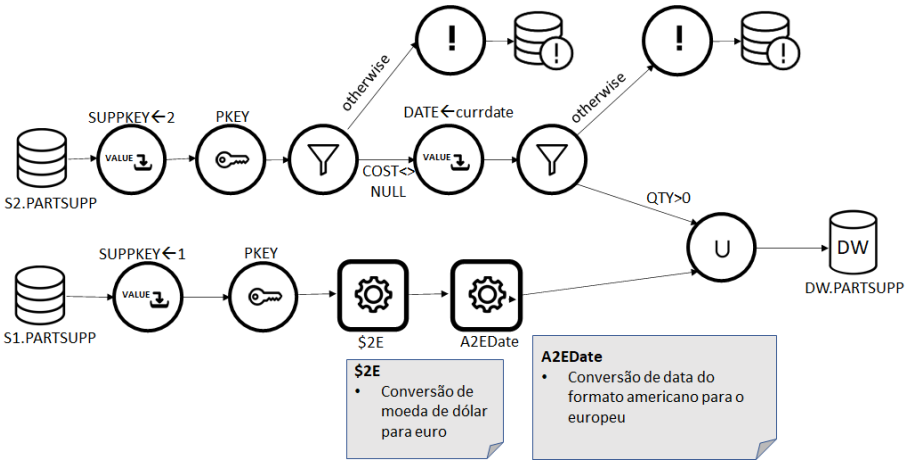



**Figura 63: Esquema conceitual com o modelo Intuitive**

Fonte: Elaborada pela autora

A primeira análise envolveu os esquemas conceituais elaborados usando o modelo Intuitive e o modelo do trabalho correlato baseado em Grafo e considerando as características de cada modelo, os seguintes resultados foram obtidos.

| | Modelo Conceitual com Grafo | Modelo Intuitive |
|---|---|---|
| Característica principal | Mapeamento dos atributos | Elementos do *workflow* de ETL |
| Construtores específicos para ETL | Sim | Sim |
| Construtores reusáveis | Sim | Sim |
| Padronização dos construtores | Sim | Sim |
| Completude | Sim, com o modelo lógico | Sim |
| Poder expressivo dos construtores | Baixo | Alto |
| Simplicidade da gramática | Baixa | Média |
| Simplicidade dos diagramas | Baixa | Alta |
| Admite adição de funções específicas | Sim | Sim |
| Necessidade de informações complementares para compreensão | Sim | Não |

**Tabela 4. Comparação entre o modelo conceitual de Grafo e o modelo Intuitive**

Na modelagem conceitual com Grafo, o foco está no tratamento dos atributos, ou seja, no mapeamento de cada atributo considerando a estrutura da fonte dos dados e do DW. No modelo Intuitive, no entanto, o foco está nos elementos do *workflow*, ou seja, na organização das operações que devem

aplicadas aos dados desde a extração até a carga no DW. Assim, para um mesmo cenário de ETL, os esquemas conceituais obtidos apresentam muitas diferenças. Por isso, o escopo da atividade de comparação foi ampliado: a comparação foi refeita considerando, adicionalmente, o modelo conceitual de Grafo em conjunto com o modelo lógico de Grafo de Arquitetura e, do outro, o modelo Intuitive. As características avaliadas foram as mesmas da comparação anterior. A Tabela 5 apresenta os resultados dessa segunda análise.

| | Modelo Conceitual com Grafo + Modelo Lógico com Grafo de Arquitetura | Modelo Intuitive |
|---|---|---|
| Característica principal | Mapeamento dos atributos | Elementos do *workflow* de ETL |
| Construtores específicos para ETL | Sim | Sim |
| Construtores reusáveis | Sim | Sim |
| Padronização dos construtores | Sim | Sim |
| Completude | Sim, com o modelo lógico | Sim |
| Poder expressivo dos construtores | Baixo | Alto |
| Simplicidade da gramática | Baixa | Média |
| Simplicidade do diagrama | Complexo, por exigir a análise simultânea de dois esquemas, com notações distintas e com elementos pouco expressivos | Simples, porque as informações estão em um único esquema que possui construtores expressivos |
| Admite adição de funções específicas | Sim | Sim |
| Necessidade de informações complementares para compreensão | O modelo lógico inclui informações detalhadas em nível menos abstrato e mais próximo da implementação. | Necessidade de detalhes sobre os atributos |

**Tabela 5. Comparação do conjunto formado por modelo conceitual de Grafo e modelo lógico de Grafo de Arquitetura com o modelo Intuitive**

No modelo Intuitive, as informações estão em um único esquema o que facilita a análise e a compreensão. No entanto, não há uma visão integrada do tratamento de cada atributo e para isso parece importante que um catálogo de dados seja incorporado ao modelo, contendo detalhes como nome, tipo e tamanho dos atributos, tanto das fontes dos dados como também do DW. Com a adição de um catálogo de dados, o modelo com operadores tem elementos suficientes para representar a mesma informação que é obtida com o conjunto formado pelo modelo conceitual por Grafo e pelo modelo lógico por Grafo de Arquitetura.

## 6.5 **Compreensão e comparação dos modelos**

A etapa de compreensão do modelo foi realizada por meio de um experimento com a participação de usuários. Foi elaborado um formulário na ferramenta GoogleForms contendo 2 seções:

- na primeira, as imagens do esquema conceitual com o uso do modelo de Grafo e com o uso do modelo Intuitive, para um mesmo cenário de ETL, foram apresentadas e os usuários foram incentivados a registrar o grau de satisfação com as seguintes características de qualidade: (i) facilidade de compreensão, (ii) representatividade semântica dos elementos, (iii) padronização dos elementos e, (iv) satisfação geral.

- na segunda seção, as imagens do esquema conceitual com o uso do modelo de Grafo, do esquema lógico com o uso do modelo de Grafo de Arquitetura e do esquema conceitual com uso do modelo Intuitive para um mesmo cenário de ETL foram apresentadas e os usuários foram incentivados a, considerando o conjunto de esquema com Grafo e com Grafo de Arquitetura em comparação com o esquema com o modelo Intuitive, registrar o grau de satisfação com as seguintes características de qualidade (as mesmas características da seção anterior: (i) facilidade de compreensão, (ii) representatividade semântica dos elementos, (iii) padronização dos elementos e, (iv) satisfação geral.

Cada seção do formulário tinha também um espaço para comentários e sugestões. O experimento foi atribuído para alunos de pós-graduação, formados em cursos de graduação na área de tecnologia: Ciência da Computação, Engenharia da Computação, Análise e Desenvolvimento de Sistemas ou outros cursos correlatos. Os participantes receberam também, como material de apoio, os artigos que descrevem cada um dos modelos e o capítulo desta dissertação que descreve o modelo Intuitive. Os resultados coletados são apresentados a seguir, juntamente com a análise e a conclusão.

Foram obtidas 11 respostas. A Figura 64 apresenta as respostas das questões objetivas que foram obtidas na primeira seção do experimento de comparação dos modelos conceituais, sem considerar o modelo lógico. As respostas refletem o grau de satisfação dos participantes com relação às características de facilidade de compreensão, representatividade semântica dos

elementos gráficos, padronização dos elementos gráficos e satisfação geral. Os resultados foram positivos para o modelo Intuitive, principalmente com relação a representatividade semântica e à padronização dos operadores.

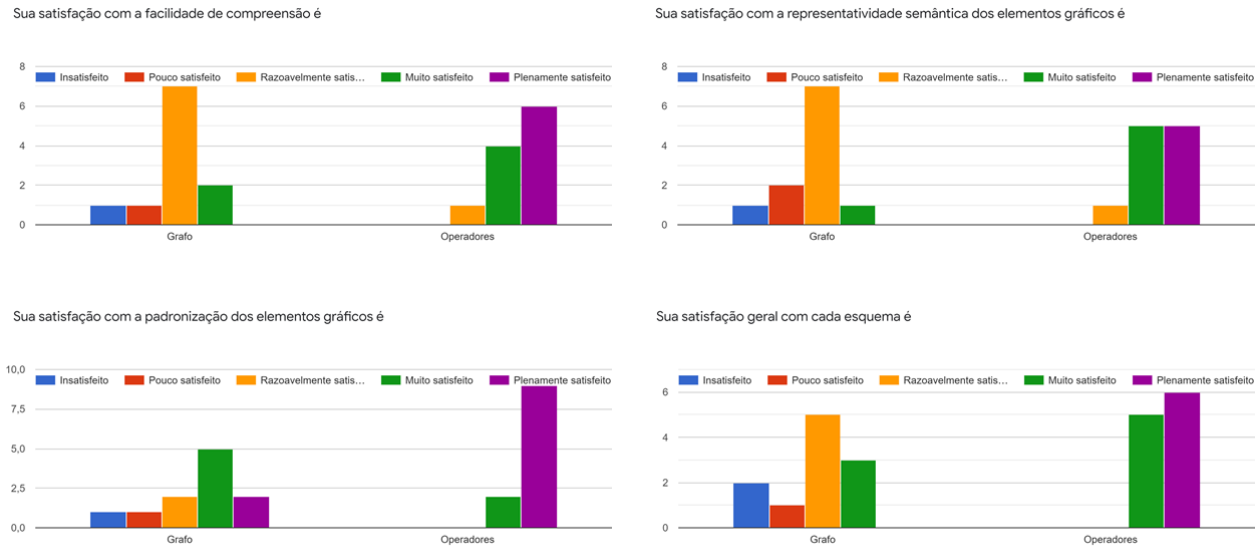



**Figura 64: Resultados obtidos na primeira parte do experimento de compreensão e comparação dos modelos Intuitive e de Grafos.**

Fonte: Elaborada pela autora

Ainda com relação à primeira seção do experimento, 4 participantes contribuíram com comentários na questão aberta: dois comentários foram referentes ao formato do experimento e os demais relataram a percepção de ser mais intuitivo trabalhar com o modelo Intuitive (daí surgiu o nome do modelo), citando que a curva de aprendizado é melhor nesse modelo. Uma contribuição interessante se refere à necessidade de ter uma visão integrada da estrutura (atributos) das fontes de dados no modelo Intuitive, sugerindo a incluir um catálogo (mapa) dos atributos no modelo proposto.

Acerca da segunda seção do experimento, referente à comparação entre o modelo Intuitive com o conjunto formado pelos modelos de Grafos e de Grafo de Arquitetura, a figura 65 apresenta as respostas das questões objetivas. As respostas refletem o grau de satisfação dos participantes com relação às características de facilidade de compreensão, representatividade semântica dos elementos gráficos, padronização dos elementos gráficos e satisfação geral. Nessa nova comparação, mais participantes declararam estar razoavelmente

satisfeitos com a facilidade de compressão dos modelos de Grafos e de Grafo de Arquitetura, demonstrando que as informações do modelo lógico complementam o modelo conceitual e contribuem para melhorar a compreensão. Ainda assim, mesmo com a inclusão do modelo lógico, as respostas apontam vantagens para o modelo Intuitive.

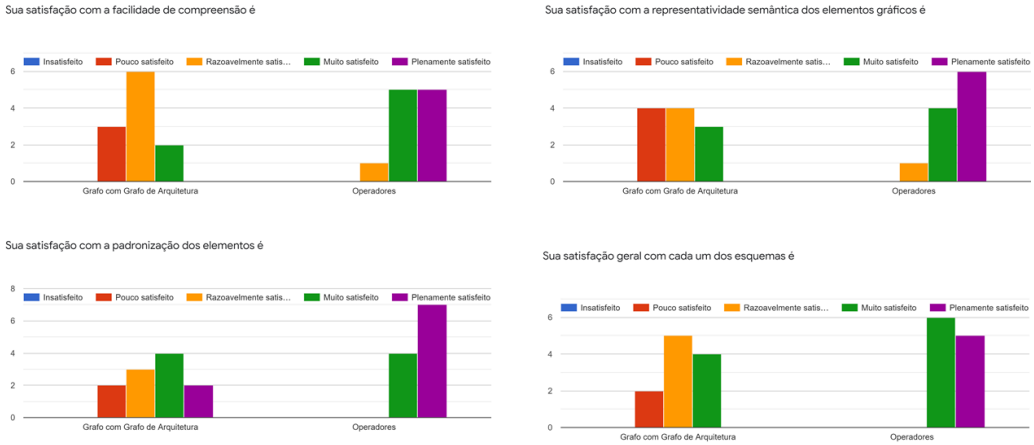



**Figura 65: Resultados obtidos na segunda parte do experimento de compreensão e comparação dos modelos Intuitive e de Grafos.**

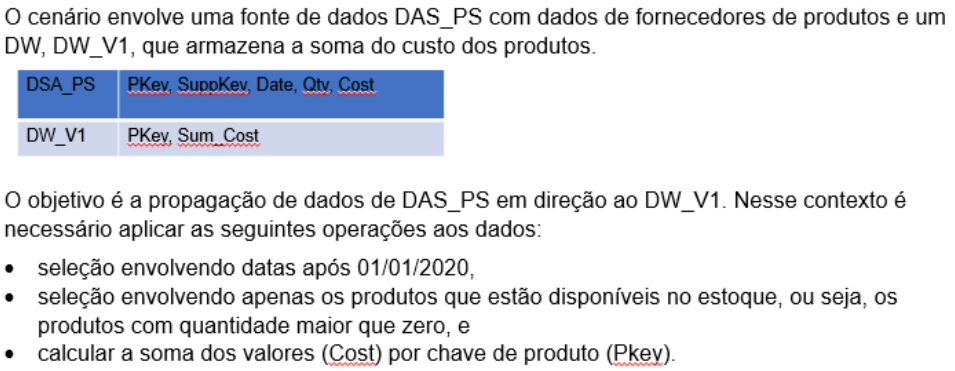
Fonte: Elaborada pela autora

Sobre a questão para resposta aberta, na segunda seção do experimento, 3 contribuições foram recebidas: dúvidas relacionadas com alguns pontos específicos de cada modelo, observação de que o conjunto formado por modelo conceitual e lógico facilitam a compreensão e de que há detalhes no modelo lógico que talvez não sejam necessários nas fases mais iniciais da construção de um DW.

## 6.6 **Aplicação do Modelo Intuitive**

Para completar a validação do modelo Intuitive foi realizado outro experimento prático, com participação de usuários (mesmos usuários do experimento descrito na Seção 6.6), de uso dos operadores na modelagem de *workflows* de ETL. Foram selecionados dois cenários: (i) cenário simples que é uma limitação do escopo do cenário descrito em Vassiliadis, Simitsis e Skiadopoulos, (2002), baseado nos padrões TPC-R/TPC-H, que trata de Fornecedores de Peças, e (ii) cenário de média complexidade, obtido de Kabiri,

Wadjinny e Chiadmi (2011) que trata de atividades de Vendas, com foco nas informações de Clientes. Esses cenários mostram a descrição do cenário 1 (Figura 66) e do cenário 2 (Figura 67) no formato que foi atribuído para os - participantes.


O cenário envolve uma fonte de dados DAS_PS com dados de fornecedores de produtos e um DW, DW_V1, que armazena a soma do custo dos produtos.

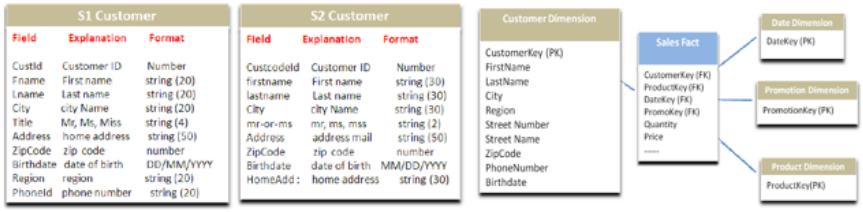
| DSA_PS | PKey, SuppKey, Date, Qtv, Cost |
|---|---|
| DW_V1 | PKey, Sum_Cost |

O objetivo é a propagação de dados de DAS_PS em direção ao DW_V1. Nesse contexto é necessário aplicar as seguintes operações aos dados:

- seleção envolvendo datas após 01/01/2020,
- seleção envolvendo apenas os produtos que estão disponíveis no estoque, ou seja, os produtos com quantidade maior que zero, e
- calcular a soma dos valores (Cost) por chave de produto (Pkey).

**Figura 66: Cenário usado no experimento de modelagem conceitual com Operadores**

Fonte: Elaborada pela autora




O cenário trata de informações de Vendas. Tem 2 fontes de dados, S1 e S2, com dados de clientes, um conjunto de dados, DB_ADRESS, que contém informações de endereços, e o DW.
O objetivo é extrair os dados de S1 e S2, tratar e carregar no DW (apenas dados da dimensão Cliente).

- Para os dados obtidos de S1, converter o formato do atributo BIRTHDATE (mm/dd/yyyy).
- Unir os dados de S1 com os dados de S2, eliminando duplicidades.
- Usar o ZipCode dos consumidores para recuperar as demais informações de endereço que estão em DB_ADRESS: City, Region e StreetName.
- Criar a chave artificial CustomerKey para identificar univocamente cada cliente.
- Atualizar os dados do DW.

**Figura 67: Cenário usado no experimento de modelagem conceitual com Operadores**

Fonte: Elaborada pela autora

Para o experimento, foi elaborado um conjunto de *slides* no MSPowerPoint contendo a notação gráfica de cada operador e a descrição de cada cenário. Os usuários foram incentivados a aplicar os operadores para modelagem conceitual de cada cenário proposto. O mesmo experimento, alternativamente, poderia ser realizado usando a ferramenta LucidChart (lucidchart.com), para a qual foi criada uma biblioteca com os operadores e um modelo de documento contendo os

operadores. A Figura 68 mostra a tela da ferramenta LucidChart com foco na biblioteca dos operadores e, do lado direito, o modelo do documento.


**Figura 68: Tela do LucidChart com a biblioteca de Operadores e o documento modelo**

Fonte: Elaborada pela autora

Recebemos respostas de 5 participantes. Aparentemente a baixa adesão está relacionada ao fato de que esse experimento exigia mais tempo e esforço para a realização. Comparamos as respostas obtidas com os gabaritos elaborados para esse experimento. De forma geral, os participantes tiveram sucesso na elaboração correta do esquema conceitual com Operadores para o primeiro cenário, do domínio de Fornecedores de Peças. As Figuras 69 e 70, retratam, respectivamente, o esquema conceitual que foi elaborado com o Intuitive e que foi a referência (gabarito) para avaliação das respostas obtidas com experimento, e os esquemas conceituais obtidos para esse cenário.

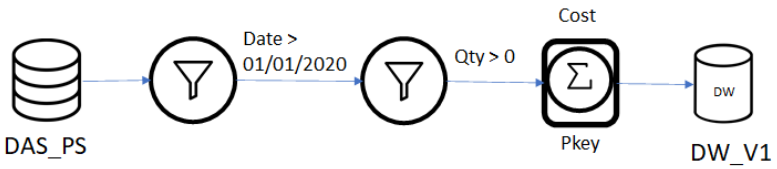



**Figura 69: Esquema conceitual com Operadores para o cenário de Fornecedores de Peças – gabarito para o experimento**

Fonte: Elaborada pela autora

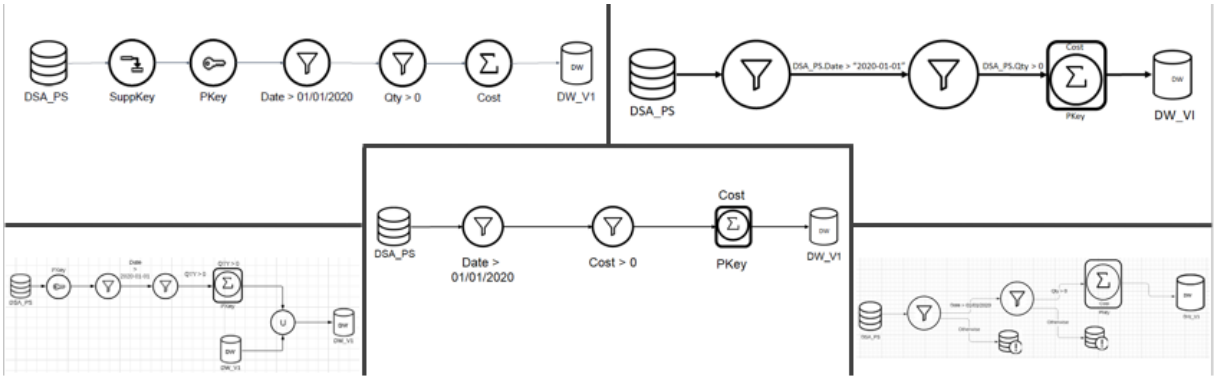



**Figura 70: Esquemas conceituais com Operadores, elaborados pelos participantes do experimento, para o cenário de Fornecedores de peças**

Fonte: Elaborada pela autora

É possível notar que o nível de detalhe contido no esquema depende do nível de conhecimento e de experiência do usuário: os participantes que têm mais conhecimento em construção de processo de ETL elaboraram esquemas mais detalhados. Não houve relatos de impedimento para a realização da atividade proposta por desconhecimento do modelo Intuitive, o que demonstra que os elementos do modelo são intuitivos e fáceis de serem compreendidos.

A Figura 71 retrata o esquema conceitual elaborado com o modelo Intuitive para o cenário de Vendas com foco em informações de Clientes e que foi a referência (gabarito) para avaliação das respostas obtidas com experimento. Os esquemas conceituais obtidos nas respostas do experimento para esse cenário estão na Figura 72.

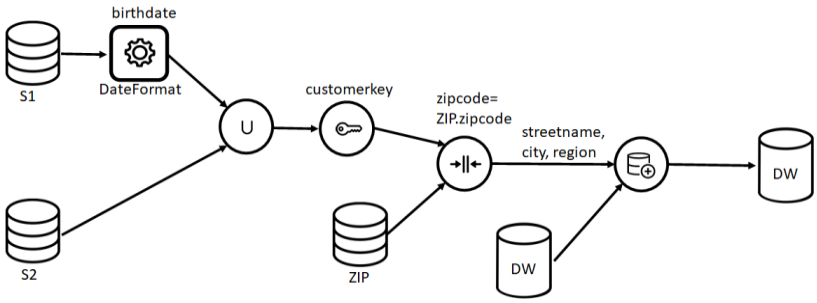



**Figura 71: Esquema conceitual com o Intuitive para o cenário de Fornecedores de Peças – gabarito para o experimento**

Fonte: Elaborada pela autora

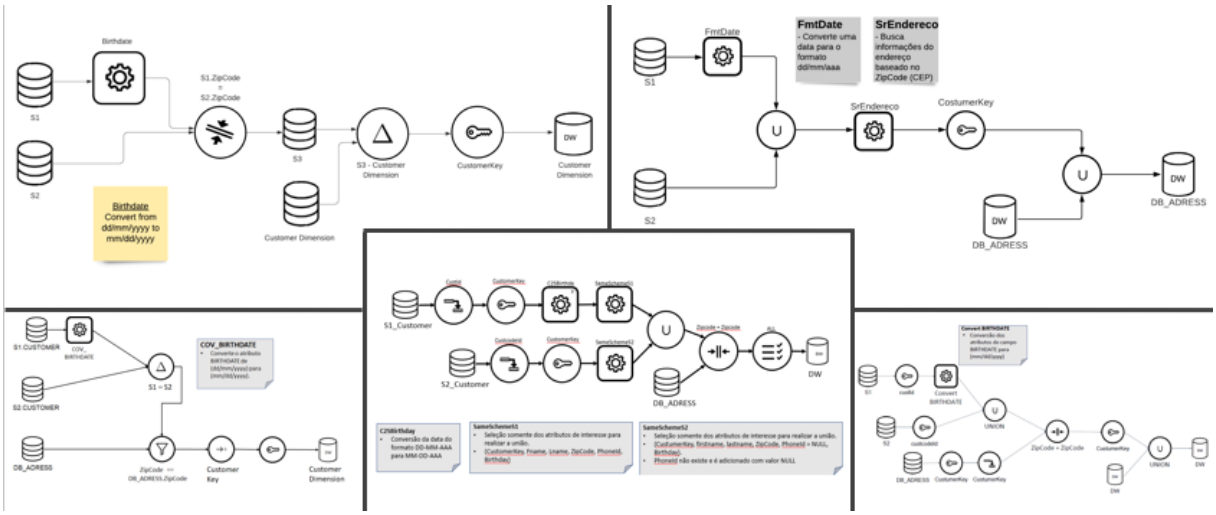



**Figura 72: Esquemas conceituais com Operadores, elaborados pelos participantes do experimento, para o cenário de Vendas com foco em Clientes**

Fonte: Elaborada pela autora

Alguns esquemas obtidos ficaram bastante completos. Novamente foi possível notar que a experiência do usuário impacta no nível de detalhes contido no esquema; apesar disso, considerando que o esquema conceitual contém as informações abstratas conhecidas no início do projeto de construção de DW, os esquemas resultantes são bastante representativos. Alguns participantes relataram que sentiram falta de mais detalhes na descrição desse segundo cenário. Em nenhum caso foi informada a impossibilidade de realizar a atividade por falta de conhecimento sobre o modelo proposto, o que indica que os operadores do modelo Intuitive são expressivos e fáceis de serem compreendidos e aplicados.

## 6.7 **Considerações finais**

Para a validação do modelo proposto foram realizadas 4 atividades: duas atividades de análise teórica e duas atividades de experimento prático com a

participação de usuários. Com isso, foi possível validar que os operadores do modelo Intuitive são suficientes para representar com clareza diversos cenários representativos de ETL e que apresenta vantagens em seu uso quando comparado ao principal trabalho do estado da arte. Alguns operadores propostos não foram usados nos cenários dessa validação e então é necessário avaliar a relevância deles. Há espaço para realização de outros experimentos práticos, envolvendo usuários que tenham mais experiência com projetos de construção de DW, em particular com o processo de ETL e com o tratamento de cenários reais e mais complexos.

# Capítulo 7

# CONCLUSÕES

*Nesse capítulo são apresentadas as conclusões dessa pesquisa de Mestrado, com destaque para as contribuições e os resultados obtidos. Além disso são indicados possíveis trabalhos futuros para continuação desta pesquisa.*

Essa pesquisa de Mestrado atingiu seu objetivo principal, ao investigar o processo de construção de ETL em ambientes de *data warehousing*, com especial atenção às fases mais iniciais desse processo, quando é realizada a modelagem do *workflow* de ETL em nível conceitual. Essa investigação possibilitou a identificação de pontos de melhoria e, nesse contexto, foi proposto o modelo Intuitive para modelagem conceitual de *workflows* de ETL, que é a principal contribuição dessa pesquisa de Mestrado.

O modelo Intuitive traz um conjunto de elementos reutilizáveis, os operadores, que representam operações frequentemente presentes em *workflows* de ETL. A notação gráfica que foi criada para os operadores tem representatividade semântica que favorece a compreensão dos esquemas conceituais podendo, assim, contribuir para melhorar a comunicação entre os usuários da área de tecnologia, ou seja, os desenvolvedores e projetistas, e os usuários não técnicos, ou seja, aqueles que têm o domínio da área de negócio e são, por assim dizer, os detentores do conhecimento sobre os requisitos de negócio que devem ser atendidos pelo DW. A melhoria na comunicação pode contribuir para o engajamento das partes interessadas no projeto e para a redução do retrabalho, podendo melhorar a produtividade e a agilidade na construção do DW. Outro benefício oferecido pelo modelo Intuitive é que os

esquemas servem para documentar as decisões tomadas durante o projeto e facilitam a análise de impacto de modificações evolutivas e corretivas e, além disso, a investigação de diversos cenários na busca de soluções mais eficientes para *workflows* de ETL.

Ao analisar as questões de pesquisa levantadas no início desse trabalho e considerando toda a investigação realizada, foi possível concluir que (i) a representação visual das operações que compõem o processo de ETL pode facilitar a construção de ambientes de *data warehousing*, (ii) o uso de elementos gráficos que sejam simples e expressivos, pode contribuir para a abstração e para a visualização do tratamento necessário para compatibilização dos dados extraídos das fontes de dados operacionais com a estrutura proposta para o DW. Além disso foi possível perceber (i) que a padronização da etapa de modelagem de processo de ETL tem potencial para contribuir no aumento da produtividade em projetos de construção de DW e, nesse sentido, é necessário ampliar o escopo dos experimentos, tratando de cenários reais de ETL com a participação de usuários experientes, (ii) como os desafios impostos pelos grandes volumes de dados e a volatilidade dos requisitos de negócio demandam frequentes manutenções nos ambientes de DW, sendo possível que os esquemas conceituais elaborados com o Modelo Intuitive facilitem a análise de impacto das mudanças.

Para a validação do modelo proposto foram realizadas 4 atividades: duas atividades de análise teórica e duas atividades de experimento prático com a participação de usuários. Com isso, foi possível validar que os operadores do modelo Intuitive foram suficientes para representar com clareza diversos cenários representativos de ETL e que seu uso apresenta vantagens quando comparado ao principal trabalho do estado da arte. Os estudos teóricos comparativos entre os modelos e o uso do Modelo Intuitive permitiram notar que a padronização da etapa de modelagem de processo de ETL pode melhorar o entendimento do *workflow* de ETL, contribuindo para aumentar a produtividade e diminuir o esforço necessário para a construção do ambiente de *data*

*warehousing*. Os sistemas de ETL são complexos e a manutenção deles exige grande esforço do desenvolvedor para análise de impacto e os estudos teóricos apontam que o modelo Intuitive pode contribuir para facilitar a análise de impacto de manutenções, mas ainda são necessários experimentos mais abrangentes e sistemáticos para comprovação.

Possíveis trabalhos complementares e futuros são (i) aumentar o escopo da validação do modelo, envolvendo experimentos com cenários reais de ETL e usuários experientes, (ii) disponibilizar uma interface gráfica que facilite a elaboração dos esquemas conceituais com o uso do modelo Intuitive, (iii) estender o modelo conceitual proposto de forma a representar atributos e as suas transformações, (iv) tratar operações de propagação de dados do DW para visões materializadas e para *data marts*; (v) estabelecer uma forma de conversão do esquema conceitual gerado usando o modelo Intuitive para ser aplicada à etapa de modelagem lógica do *workflow* de ETL, tendo como alvo ferramentas específicas com ênfase em ferramentas *open source* (tal como Kettle da Pentaho, Talend Open Studio e CloverETL) e (vi) investigar a otimização do esquema conceitual em termos de parâmetros de custo mais abstratos para formas alternativas de *workflows* de ETL para um mesmo cenário.

# REFERÊNCIAS


ALI, S. M. F.; WREMBEL, R. **From conceptual design to performance optimization of ETL workflows: current state of research and open problems**, The VLDB Journal 26: 777, 2017.

BALA, M.; BOUSSAID, O.; ALIMAZIGH, Z. **Big-ETL: Extracting-Transforming-Loading Approach for Big Data**. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), 2015, 462 p.

BRITO, J. J. **Data Warehouses in the era of Big Data: efficient processing of Star Joins in Hadoop**. 2018 p. Tese (Doutorado em Ciências – Ciências da Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

CHAKRABORTY, J.; PADKI, A.; BANSAL S. K. **Semantic ETL – State-of-the-art and open research challenges**, 2017 IEEE 11th International Conference on Semantic Computing (ICSC), San Diego, CA, 2017. p. 413-418.

CHAUDHURI, S., DAYAL, U. **An Overview of Data Warehousing and OLAP Technology,** in ACM Sigmod Record, 1997.

CHEN, M.; MAO, S.; LIU, Y. **Big Data: A Survey**. Mobile Networks Applications, 2014. 171-209.

CIFERRI, C. D. A. **Distribuição de dados em ambientes de data warehousing: o sistema WebD2W e algoritmos voltados à fragmentação horizontal dos dados**. Tese (Doutorado em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife, 2002.

EL-SAPPAGH, S. H. A., HANDAWI, A.M. A., EL BASTAWISSY, A. H. **A proposed model for data warehouse ETL processes**, Journal of King Saud University – Computer and Information Sciences, 23, 2011. p. 91–104.

EL AKKAOUI, Z., ZIMÁNYI, E. **Defining ETL Workflows Using BPMN and BPEL,** In: Proceedings of ACM International Workshop on Data Warehousing and OLAP (DOLAP'09), Hong Kong, China, 2009.

EL AKKAOUI, Z., ZIMÁNYI, E., MAZÓN J.; TRUJILLO J. **A Model-Driven Framework for ETL Process Development,** In: Proceedings of ACM International Workshop on Data Warehousing and OLAP (DOLAP'11), Glasgow, Scotland, 2011.

ELMASRI, R., NAVATHE, S. B. **Sistemas de Banco de Dados***;* tradução Daniel Vieira; 6ª edição. Pearson Addison Wesley, 2011.

GEMINO, A., WAND, Y. **A framework for empirical evaluation of conceptual modeling techniques**, Requirements Eng 9, 2004. p. 248–260.

GOLFARELLI, M. **From User Requirements to Conceptual Design in Data Warehouse Design**, 2010.

INMON, W. H. **Building the Data Warehouse**. 4 ed. Indianápolis, Estados Unidos da América, Willey Computer Publishing, 2005. 443 p.

INMON, W. H.; STRAUS, D.; NEUSHLOSS, G. **DW 2.0 – The Architecture for the Next Generation of Data Warehousing**, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

KABIRI, A.; WADJINNY, F.; CHIADMI, D. **Towards a Framework for Conceptual Modeling of ETL Processes**. In: Proceedings of the first international conference on Innovative Computing Technology, INCT 2011. Communications in Computer and Information Science, Springer Berlin Heidelberg, 2011. vol 241.

KABIRI, A.; CHIADMI, D. **A method for modelling and organizing ETL Processes**. In: Proceeding of the second international conference on Innovative Computing Technology, INTECH 2012, Casablanca, 2012. p.138-143.

KABIRI, A.; CHIADMI, D. **Survey on ETL Processes**. In: Journal of Theoretical and Applied Information Technology, 2013. vol. 53.

KIMBALL, R.; ROSS, M. **The data warehouse toolkit: the complete guide to dimensional modeling**. 2 ed. United States of America: Wiley Computer Publishing, 2002, 436 p.

KIMBALL, R.; CASERTA, J. **The data warehouse ETL toolkit: practical techniques for extracting, cleaning, conforming, and delivering data**. United States of America: Wiley Computer Publishing, 2004, 491 p.

KOUGKA, G.; GOUNARIS, A.; SIMITSIS, A. **The Many Faces of Data-centric Workflow Optimization: A Survey**, 2017.

LIU, X., THOMSEN, C., PEDERSEN T. **ETLMR: A Highly Scalable Dimensional ETL Framework Based on MapReduce**. In: Data Warehousing and Knowledge Discovery, Springer, 2011. p. 96–111.

MEHMOOD, K.; CHERFI, S. S., COMYN-WATTIAU I. **Data Quality Through Conceptual Model Quality – Reconciling Researchers and Practitioner through a Customizable Quality Model,** DBLP, 2009. Disponível em: www.reasearchgate.net

MOODY, D. L. **Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions,** Data & Knowledge Engineering 55, 2005. p. 243–276. Disponível em: www.sciencedirect.com.

MUKHERJEE R.; KAR P. **A Comparative Review of Data Warehousing ETL Tools with New Trends and Industry Insight**, 2017 IEEE 7th International Advance Computing Conference (IACC), 2017. p. 943-948.

PAN, B.; ZHANG, G.; QIN, X. **Design and Realization of an ETL Method in Business Intelligence**. 3rd IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Chengdu, 2018. p. 275-279.

SIMITSIS, A.; VASSILIADIS, P.; SELLIS, T. **Extraction-Transformation-Loading Processes**, Encyclopedia of Database Technologies, and Applications, IGI Global, 2005. p. 240-245.

SIMITSIS, A., VASSILIADIS, P. **A methodology for the conceptual modeling of ETL processes**, Proceedings of the of Conference on Advanced Information Systems Engineering (CAiSE) (2003), 2003.

SIMITSIS, A., **Modeling and managing ETL processes**, VLDB PhD Workshop, 2003.

SIMITSIS, A., SKOUTAS, D., CASTELLANOS, M. **Representation of conceptual ETL designs in natural language using Semantic Web technology**, Data & Knowledge Engineering, 2010. vol 69, p. 96-115.

SIMITSIS, A., WILKINSON K., CASTELLANOS, M., DAYAL, U. **QoX-driven ETL design: reducing the cost of ETL consulting engagements**, Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (SIGMOD '09), 2009.

SKOUTAS, D.; SIMITSIS, A. **Ontology-based Conceptual Design of ETL Processes for both Structured and Semi-structured Data**. International Journal Semantic Web Inf. Syst. (IJSWIS), 2007. p.1–24,

THOMSEN, C., PEDERSEN, T. B. **pygrametl: A Powerful Programming Framework for Extract–Transform–Load Programmers**, Proceedings of ACM International Workshop on Data Warehousing and OLAP (DOLAP'09), Hong Kong, China, 2009.

TRUJILLO, J., LUJÁN-MORA, S. **A UML based approach for modeling ETL processes in data warehouses**. Proceedings of International Conference on Conceptual Modeling (ER). Springer Berlin, 2003.

SIMITSIS, A., VASSILIADIS, P., DAYAL U., KARAGIANNIS, A., TZIOVARA V. **Benchmarking ETL Workflows**. In: Performance Evaluation and Benchmarking. TPCTC Lecture Notes in Computer Science, Springer Berlin, Heidelberg, 2009. vol 5895.

VASSILIADIS, P. **A Survey on Extract-Transform-Load Technology**, International Journal of Data Warehousing and Mining, 2009.

VASSILIADIS, P; SIMITSIS, A.; BAIKOUSI, E. **A taxonomy of ETL activities**, Proceedings of the ACM 12º International Workshop on Data Warehousing and OLAP (DOLAP '09). ACM, New York, NY, USA, 2009. p. 25-32.

VASSILIADIS, P., SIMITSIS, P., SKIADOPOULOS, A. **Conceptual modeling for ETL processes**, Proceedings of ACM International Workshop on Data Warehousing and OLAP (DOLAP'02), 2002.

VASSILIADIS, P., SIMITSIS, P., SKIADOPOULOS, A. **Modeling ETL activities as graphs**, In: Proceedings of the 4th International Workshop DMDW'2002, Toronto, Canada, 2002. p. 52-61.

VASSILIADIS, P., TERROVITIS, M., SKIADOPOULOS, **A Blueprints and Measures for ETL Workflows**. International Conference on Conceptual Modeling, ER 2005, Klagenfurt, Austria, 2005. p. 24-28.

VASSILIADIS, P., VAGENA, Z. SKIADOPOULOS, S., KARAYANNIDIS, N., SELLIS, T. **Arktos: Towards the Modeling, Design, Control and Execution of ETL Processes**, Information Systems InfoSys, 26, 2001. p. 537-561.

WILKINSON, K., SIMITSIS, A., DAYAL, U., CASTELLANOS, M, **Leveraging Business Process Models for ETL Design.** In: Conceptual Modeling – ER 2010. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2010. vol 6412.