

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

**MODELO INTUITIVE: MODELAGEM CONCEITUAL
BASEADA EM OPERADORES**

AUTORA: ANA CÉLIA RIBEIRO BIZIGATO PORTES

ORIENTADOR: PROF. DR. RICARDO RODRIGUES CIFERRI

COLABORADORA: PROF. DR. CRISTINA DUTRA DE AGUIAR

São Carlos - SP

Outubro/2020

Capítulo 5

MODELO INTUITIVE: MODELAGEM CONCEITUAL BASEADA EM OPERADORES

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 operadores que permite representar as operações realizadas em um processo de ETL. A notação gráfica permite a especificação do processo de ETL logo no início do projeto de data warehousing e, com isso, facilita o entendimento e a comunicação entre os usuários e os desenvolvedores, contribuindo para a correta captura dos requisitos do processo de ETL e sua posterior documentação. Por ser independente de tecnologia e por usar notação visual simples e representativa, 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 obtenção de resultados que agreguem 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 do processo de ETL, mas, segundo Ali e Wrembel (2017), essas abordagens ainda exigem significativo esforço dos projetistas para a construção de um *workflow* de ETL, pois 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 apresentado o modelo Intuitive, o qual é voltado para a modelagem conceitual de *workflows* de ETL, utiliza um conjunto de operadores conjuntamente com uma notação gráfica simples e representativa. Os operadores propostos visam agilizar o trabalho das fases iniciais do projeto de DW e, com isso, facilitar a integração entre usuários não técnicos, por exemplo usuários da área de negócios, e a equipe técnica envolvida nesses projetos. O modelo Intuitive aplicado a processos de ETL em DW é detalhado nas próximas seções.

5.2 Operadores

No modelo Intuitive, as tarefas de ETL são representadas por operadores que são construtores com alto nível de abstração. Para representar o *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 é um ou mais repositórios que representam as fontes de dados e, de forma semelhante, o final é um ou mais repositórios, sendo que o principal é o DW. Assim, os operadores e os relacionamentos constituem uma linguagem visual que permite a representação abstrata das sucessivas operações de extração, transformação, limpeza e carga que são aplicadas aos dados no processo de ETL.

Cada operador tem uma notação gráfica específica, sendo definido por entradas, parâmetros e saídas. As entradas podem 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. Os parâmetros podem ser (i) nomes de atributos dos conjuntos 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, as saídas, que são conjuntos de dados direcionados para repositórios ou para outras tarefas de ETL, podem 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.

Os operadores propostos podem ser classificados em categorias, considerando as características e os efeitos que causam sobre os dados ou sobre a organização do processo: (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.

5.2.5 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 ser (i) o ponto de início de um *workflow* 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* 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, ou seja, em um *workflow* de ETL, entre dois operadores de armazenamento sempre é necessário ter pelo menos um operador. Os operadores de armazenamento propostos são DataSet, TempDataSet, FailDataSet, DataLake, DataMart e DataWarehouse.

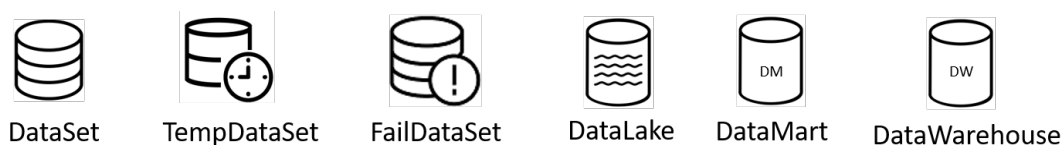


Figura 10: Notação gráfica dos operadores de armazenamento: DataSet, TempDataSet, FailDataSet, DataLake, DataMart e DataWarehouse

Fonte: Elaborada pela autora

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

Operador DataWarehouse

O operador DataWarehouse é o grande repositório onde os dados tratados são carregados e armazenados para as consultas gerenciais: os dados do DW são resultantes da aplicação de sucessivas operações de transformação e de limpeza. 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 popular 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.

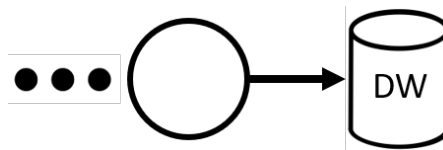


Figura 11: Cenário de uso do operador DataWarehouse

Fonte: Elaborado pela autora

Operador DataMart

No 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 ou, em alguns ambientes de *data warehousing*, os dados do DW podem ser direcionados para construir *data marts* que são usados para as consultas gerenciais. Assim, na representação conceitual do *workflow* de ETL, a entrada e a saída para o operador DataMart são ambas unárias e não obrigatórias.

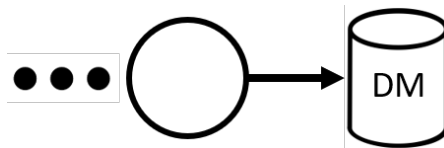


Figura 12: Cenário de uso do operador DataMart

Fonte: Elaborado pela autora

A figura 12 ilustra o uso do operador DataMart com um conjunto de dados de entrada resultante do processamento alguma tarefa de ETL, possivelmente alguma agregação e, nesse caso, como nenhuma saída está representada, os dados contidos no *data mart* serão consumidos para 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 semi-estruturados. De forma geral, os dados do *data fake* são processados apenas quando a informação é necessária para responder alguma consulta 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 *Data Staging Area* (DAS) para posterior carga de dados no DW. Assim, a entrada e a saída do operador DataLake são ambas unárias.

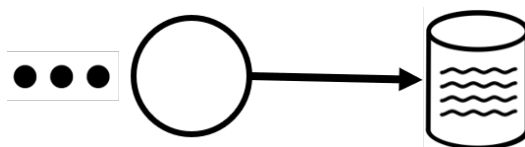


Figura 13: Cenário de uso do operador DataLake

Fonte: Elaborado pela autora

A figura 13 ilustra o uso do DataLake com um conjunto de dados de entrada que resultam do processamento de alguma tarefa 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 *Data Stage Area* (DSA).

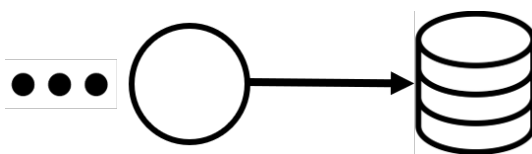


Figura 14: Cenário de uso do operador DataSet

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.

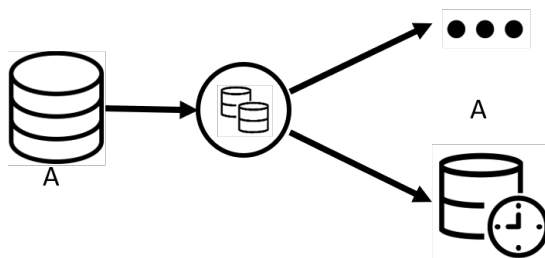


Figura 15: Cenário de uso do operador TempDataSet

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. Esse operador representa uma área de armazenamento de dados rejeitados por uma operação ou um *log* de execução, como um ponto de término do *workflow* de ETL.

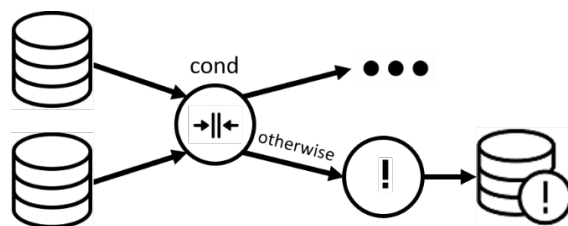


Figura 16: Cenário de uso do operador FailDataSet

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.

5.2.1 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.

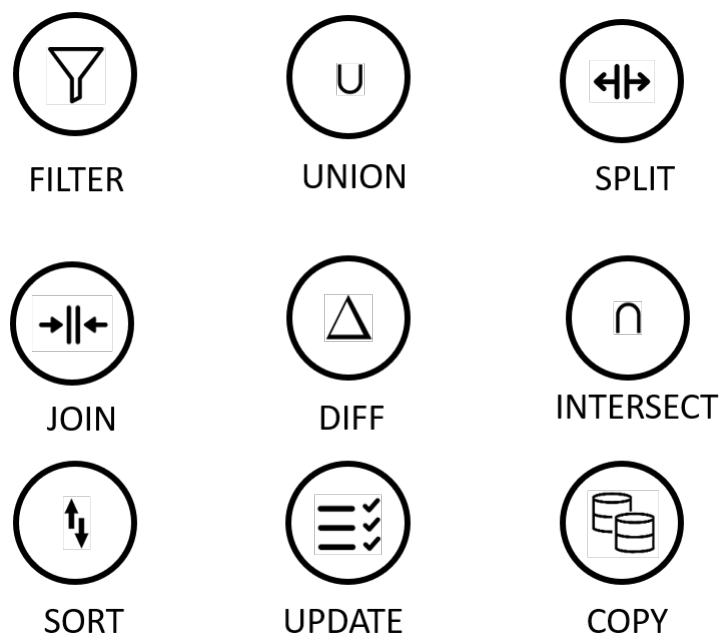


Figura 17: Operadores de manipulação de dados: Filter, Union, Split, Join, Diff, Intersect, Sort, Update e Copy.

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.

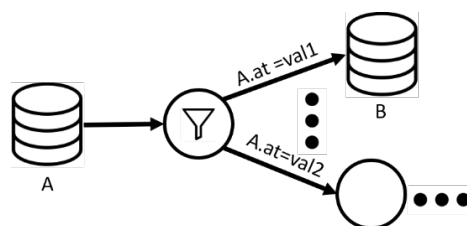


Figura 18: Cenário de uso do operador Filter.

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 tarefa 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.

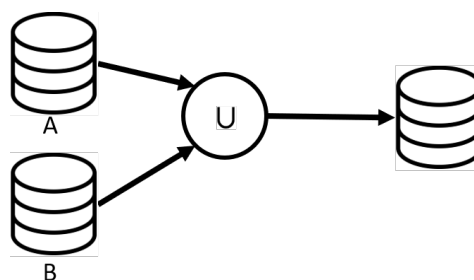


Figura 19: Cenário de uso do operador Union.

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, sem duplicidades.

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 que já haviam sido 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 forma implícita, os itens da primeira fonte de dados são priorizados e então somente os itens da segunda fonte que não têm um correspondente na primeira, são adicionados ao resultado.

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 de dados 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 mais de um subconjunto do resultado. Os dados resultantes da operação Split podem ser direcionados para outras operações do processo de ETL ou para um repositório.

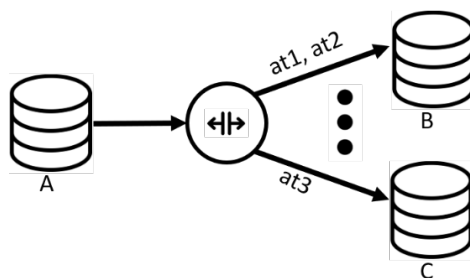


Figura 20: Cenário de uso do operador Split

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, com os atributos – at1, at2 e at3; o operador Split é aplicado para separar esses atributos, originando um novo conjunto contendo os atributos at1 e at2 e outro conjunto contendo somente o atributo at3. 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.

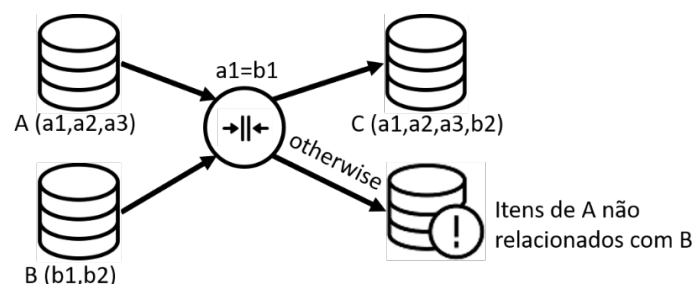


Figura 21: Cenário de uso do operador Join

Fonte: Elaborado pela autora

A Figura 21 tem a notação gráfica do operador Join em um cenário onde a entrada é formada pelos conjuntos de dados A, com os atributos a1, a2 e a3 e B, com os atributos b1 e b2 e pela 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 de dados 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 tarefas anteriores aplicadas aos dados devem estar terminadas para que o cálculo das diferenças seja iniciado.

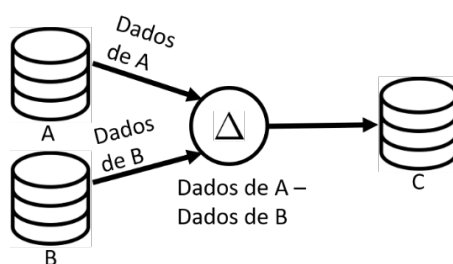


Figura 22: Cenário de uso do operador Diff

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 que estão presentes, também, no demais conjuntos, e pode também ser binária quando os itens que não atendem a condição especificada são direcionados para um fluxo alternativo {alterar a figura abaixo para incluir o fluxo alternativo e explica-lo}. 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 tarefas anteriores aplicadas aos dados devem estar terminadas para que o cálculo da intersecção seja iniciado.

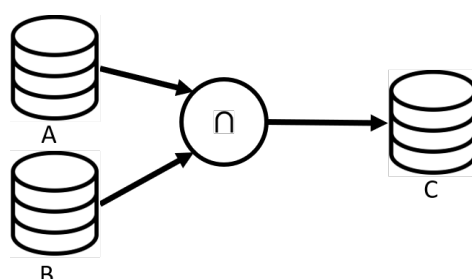


Figura 23: Cenário de uso do operador Intersect

Fonte: Elaborado pela autora

A Figura 23 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.

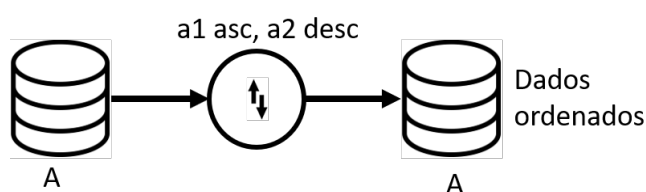


Figura 24: Cenário de uso do operador Sort

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 asc e a2 com critério de ordenação desc, e a saída que tem é 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 de forma crescente por Data de forma crescente e por Valor do Pedido de forma decrescente.

Operador Update

O operador Update representa a alteração ou atualização 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: $at1 \leftarrow val1$) 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.

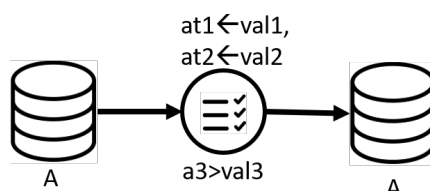


Figura 25: Cenário de uso do operador Update

Fonte: Elaborado pela autora

No cenário da Figura 25, os valores dos atributos $at1$ e $at2$ são substituídos por $val1$ e $val2$, respectivamente, para os itens onde o valor do atributo $at3$ é maior que $val3$. Supondo que A contém dados de Pedidos, $a1$, $a2$ e $a3$ são Desconto, Frete e Valor dos pedidos e que $val1$, $val2$ e $val3$ 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 24 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.

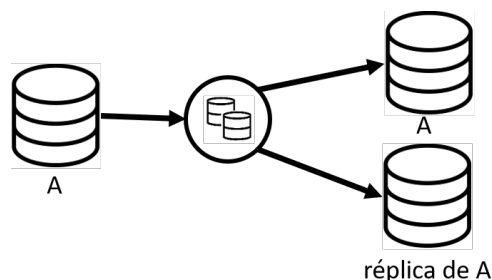


Figura 26: Cenário de uso do operador Copy

Fonte: Elaborado pela autora

5.2.2 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 um único valor como resultado. 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 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.

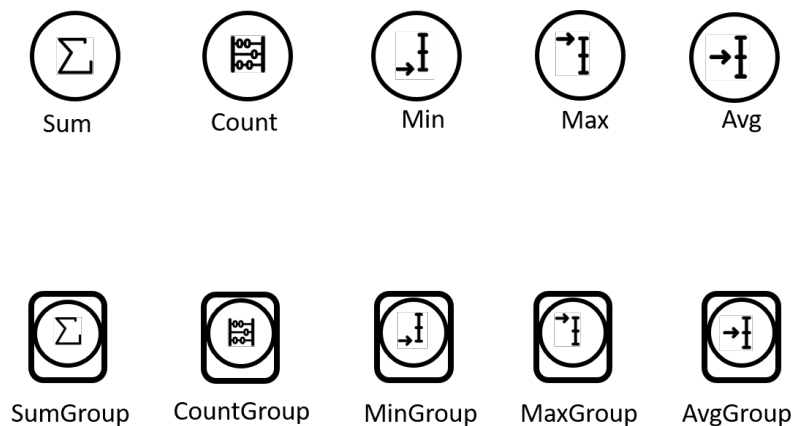


Figura 27: Notação gráfica dos operadores Sum, Count, Min, Max, Avg e os correspondentes SumGroup, CountGroup, MinGroup, MaxGroup e AvgGroup

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 são considerados na soma.

A Figura 28 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 satisfaçam 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.

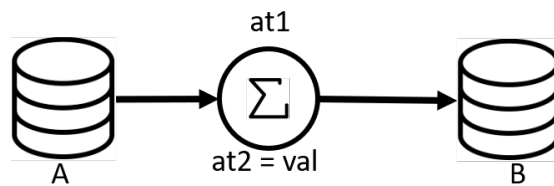


Figura 28: Cenário de uso do operador Sum.

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.

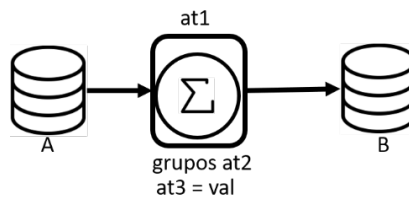


Figura 29: Cenário de uso do operador SumGroup

Fonte: Elaborada pela autora

No exemplo da Figura 29, 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.

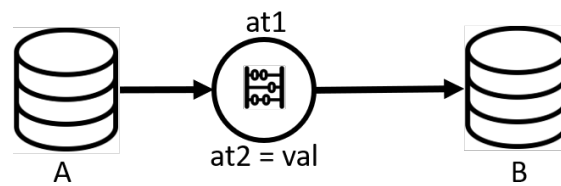


Figura 30: Cenário de uso do operador Count

Fonte: Elaborada pela autora

No exemplo apresentado na Figura 30, 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 onde Desconto é 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.

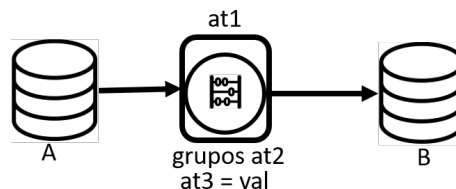


Figura 31: Cenário de uso do operador CountGroup

Fonte: Elaborada pela autora

Assim, na Figura 31, 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 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.

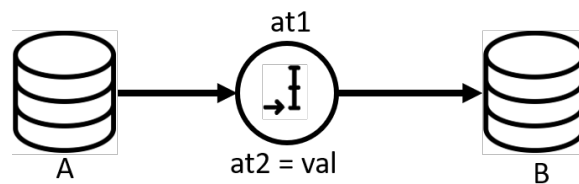


Figura 32: Cenário de uso do operador Min

Fonte: Elaborada pela autora

Considerando o exemplo retratado na Figura 32 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 zero.

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

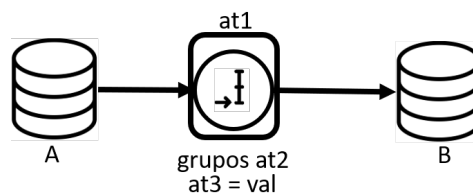


Figura 33: Cenário de uso do operador MinGroup

Fonte: Elaborada pela autora

No exemplo da Figura 33, 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 específico 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.

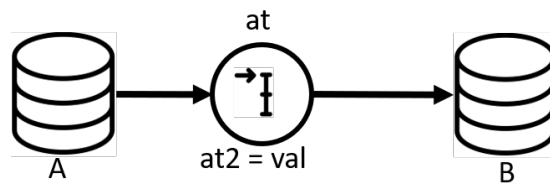


Figura 34: Cenário de uso do operador Max

Fonte: Elaborada pela autora

Considerando o exemplo retratado na Figura 34 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 como parâmetro uma lista de atributos que são usados para a formação de grupos.

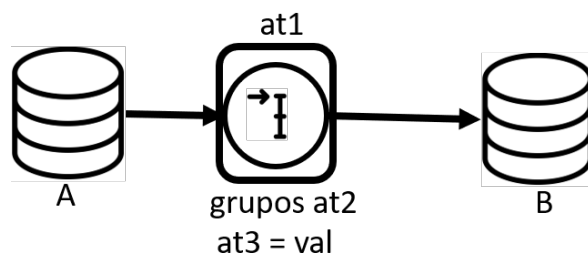


Figura 35: Cenário de uso do operador MaxGroup

Fonte: Elaborada pela autora

No exemplo da Figura 35, 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.

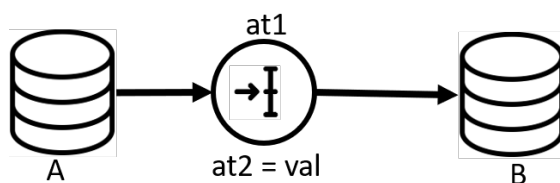


Figura 36: Cenário de uso do operador Avg

Fonte: Elaborada pela autora

Considerando o exemplo retratado na Figura 36, 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.

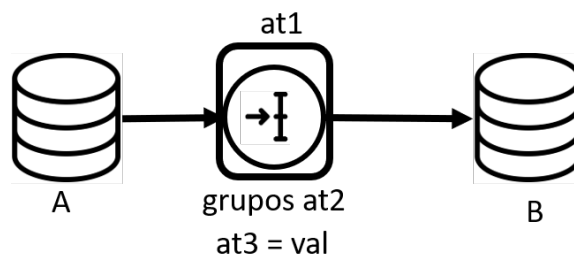


Figura 37: Cenário de uso do operador AvgGroup

Fonte: Elaborada pela autora

No exemplo da Figura 37, 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 0.

5.2.3 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 38 traz a notação gráfica dos operadores de inicialização e a descrição desses operadores é apresentada nas próximas seções.

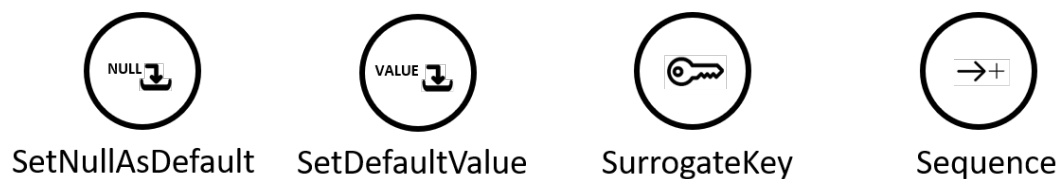


Figura 38: Notação gráfica dos operadores de inicialização

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 existem então podem ser criados e inicializados.

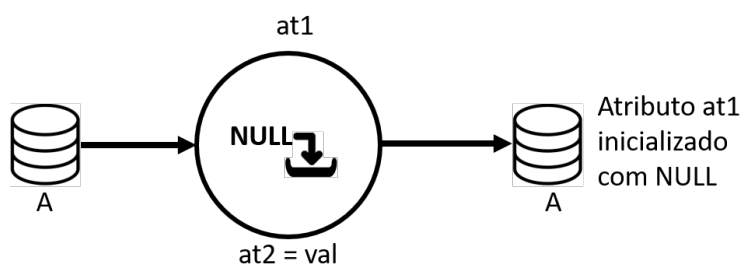


Figura 39: Cenário de uso do operador SetNullAsDefault

Fonte: Elaborada pela autora

No exemplo da Figura 39, 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 onde Desconto é 0.

Operador SetDefaultValue

O operador SetDefaultValue é usado para representar a inicialização de atributos de um conjunto de dados 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 existem, então podem ser criados e inicializados.

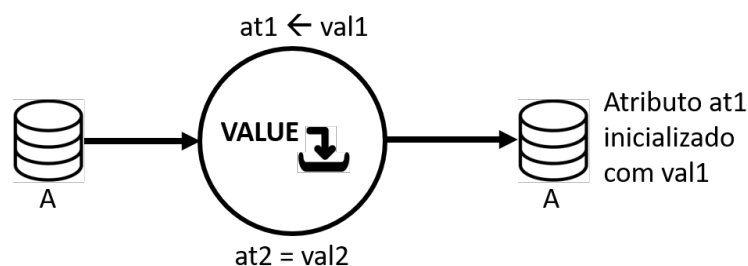


Figura 40: Cenário de uso do operador SetDefaultValue

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 se 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.

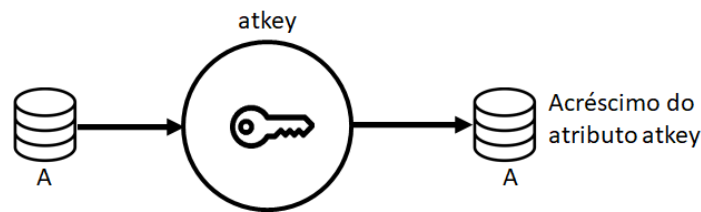


Figura 41: Cenário de uso do operador SurrogateKey

Fonte: Elaborada pela autora

A Figura 41 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 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.

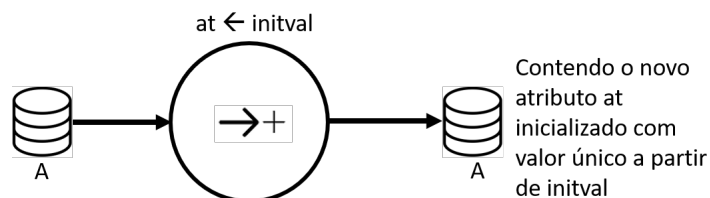


Figura 42: Cenário de uso do operador Sequence

Fonte: Elaborada pela autora

A Figura 42 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, modificado 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.

5.2.4 Operadores de fluxo

Os operadores de fluxo de dados permitem representar uma alteração 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, Sincronize, Delay e Fail. A Figura 43 ilustra a notação gráfica proposta para os operadores de fluxo e a descrição desses operadores é apresentada na próximas seções.

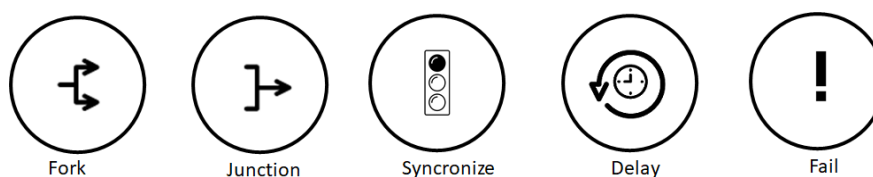


Figura 43: Notação gráfica dos operadores de fluxo: Fork, Junction, Sincronize, Delay e Fail

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.

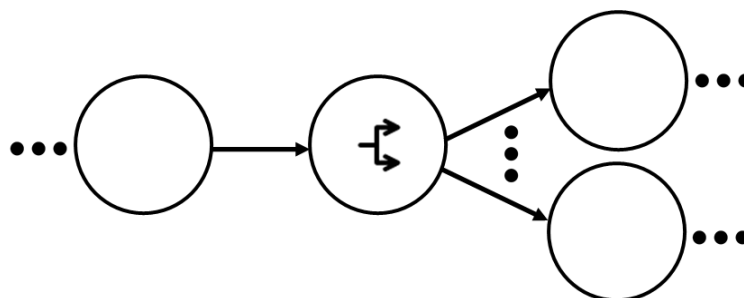


Figura 44: Cenário de uso do operador Fork

Fonte: Elaborada pela autora

A Figura 44 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*. O operador Fork é especialmente útil para criar um ponto de recuperação, permitindo que, caso ocorra alguma interrupção ou falha no processamento, seja possível retomá-lo a partir do ponto em que os dados foram salvos.

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.

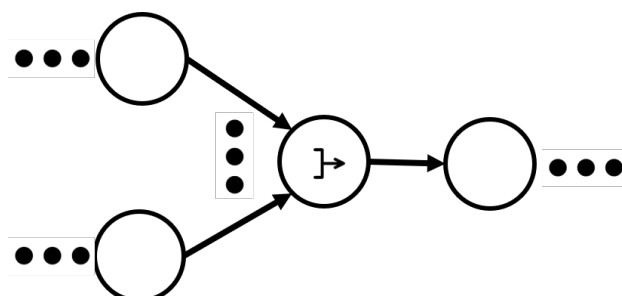


Figura 45: Cenário de uso do operador Junction

Fonte: Elaborada pela autora

A Figura 45 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 Sincronize

O operador Sincronize 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 Sincronize 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 Sincronize é n-ária (pelo menos binária) e a saída é unária. Como parâmetros, o operador Sincronize é obrigatório 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 Sincronize 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.

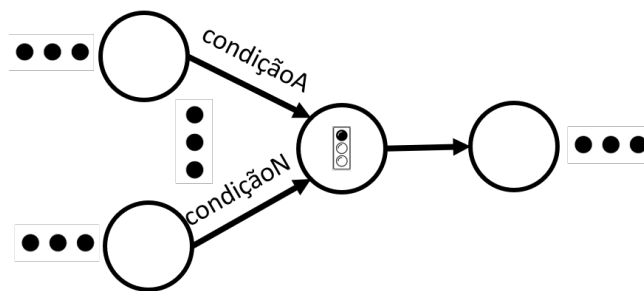


Figura 46: Cenário de uso do operador Sincronize

Fonte: Elaborada pela autora

A Figura 46 representa a aplicação do operador Sincronize com vários fluxos de entrada, uma condição para cada 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 de cada fluxo 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.

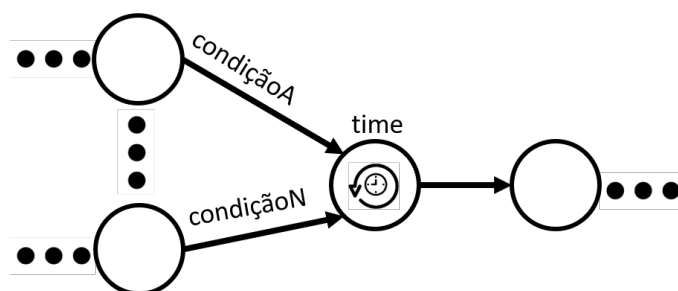


Figura 47: Cenário de uso do operador Delay

Fonte: Elaborada pela autora

A Figura 47 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 ser direcionada para alguma outra operação ou para um repositório que pode ser, por exemplo, um log de erros.

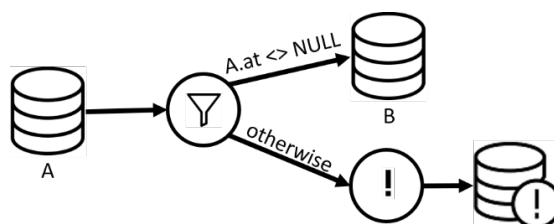


Figura 48: Cenário de uso do operador Fail

Fonte: Elaborada pela autora

A Figura 41 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 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 49.



Figura 49: 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 processo 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 processo 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.

O operador Function tem um título, que indica brevemente qual é a operação implementada e um catálogo descritivo contendo mais detalhes: 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.

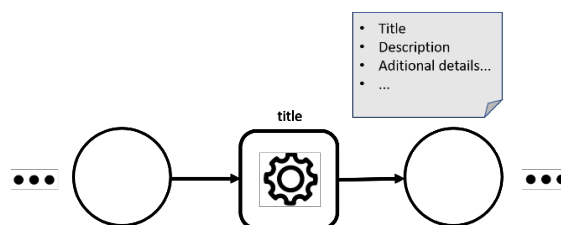


Figura 50: Cenário de uso do operador Function e catálogo descritivo

Fonte: Elaborado pela autora

A figura 47 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 outra tarefa de ETL, 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 tarefas 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.

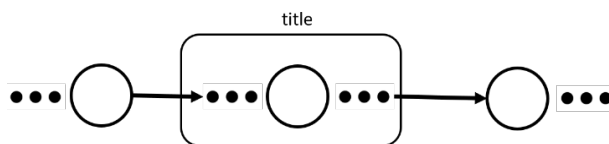


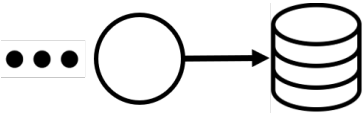
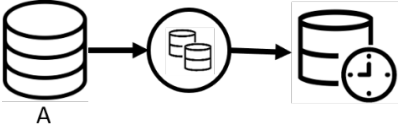
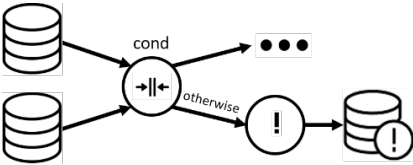
Figura 51: Cenário de uso do operador Function e catálogo descritivo

Fonte: Elaborado pela autora

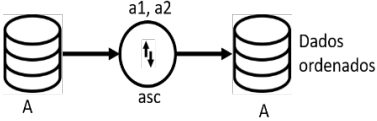
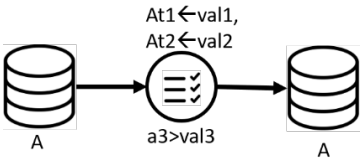
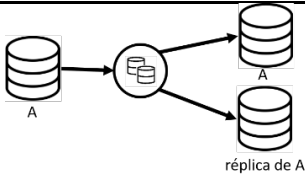
A figura 51 tem ilustra o uso do operador SubFlow com uma entrada e uma saída.

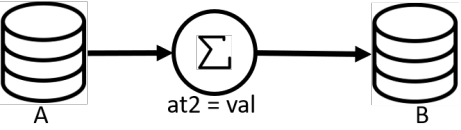
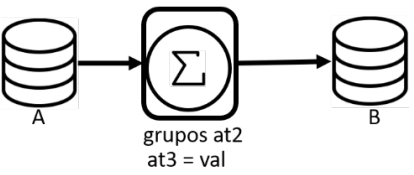
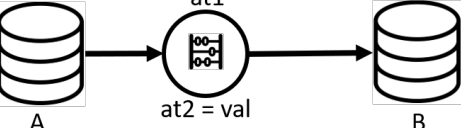
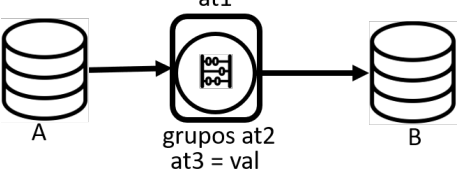
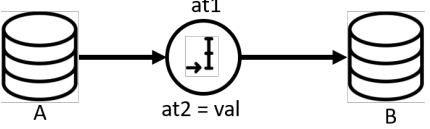
A tabela a seguir apresenta uma síntese dos operadores propostos no modelo Intuitive, contendo o nome, funcionalidade, entradas, parâmetros e saídas.

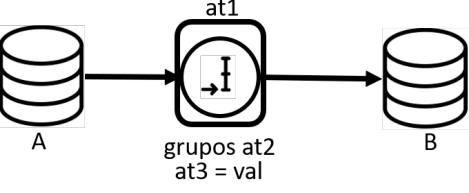
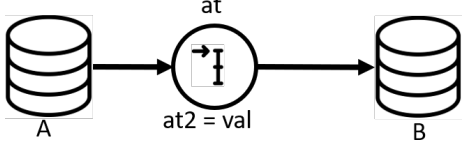
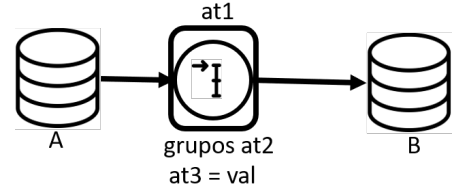
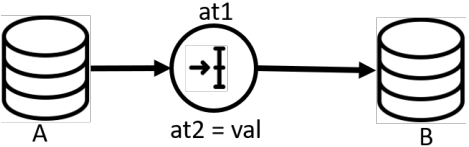
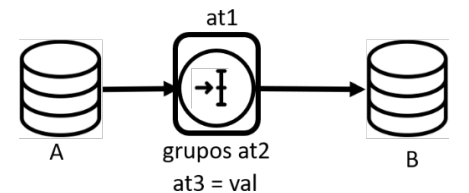
Operadores de Armazenamento			
Operadores	Funcionalidade	Notação gráfica	Entrada, Parâmetros e Saída
DataWarehouse	Armazenamento de dados tratados		Entrada: Unária e não obrigatória – conjunto de dados tratados em tarefas anteriores Saída: Unária e não obrigatória – conjunto de dados tratados para agregação, segmentação e consultas.
DataMart	Armazenamento de dados segmentados		Entrada: Unária e não obrigatória – subconjunto de dados do DW. 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. Saída: Unária e não obrigatória – dados brutos para análise.

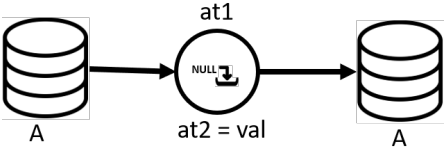
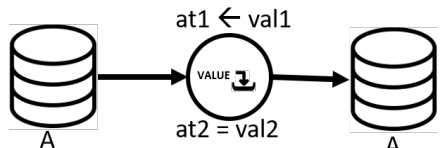
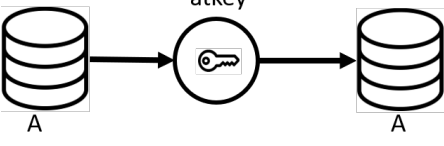
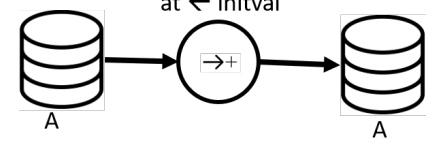
Operadores de Armazenamento			
Operadores	Funcionalidade	Notação gráfica	Entrada, Parâmetros e Saída
DataSet	Área de armazenamento tal como repositório, base da dados, tabelas ou arquivos		<p>Entrada: Unária e não obrigatória – qualquer conjunto de dados resultante da aplicação de tarefas de ETL, fontes de dados, armazenamento intermediário ou final.</p> <p>Saída: Unária e não obrigatória – qualquer conjunto de dados para tarefas de ETL.</p>
TempDataSet	Armazenamento temporário de dados		<p>Entrada: Unária e não obrigatória – qualquer conjunto de dados resultante da aplicação de tarefas de ETL, armazenados temporariamente para facilitar o processamento.</p> <p>Saída: Unária e não obrigatória – dados temporários que apoiam o processamento de alguma tarefa de ETL.</p>
FailDataSet	Armazenamento de dados rejeitados ou log de erros		<p>Entrada: Unária – dados resultantes da aplicação de alguma tarefa de ETL que tenham sido rejeitados por não atenderem a alguma condição.</p> <p>Saída: Unária e não obrigatória – dados resultantes de um fluxo alternativo para alguma tarefa de ETL.</p>

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

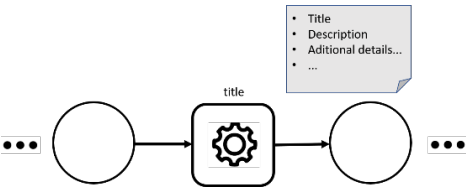
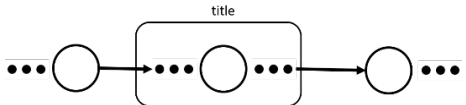
Operadores de Manipulação de Dados			
Operadores	Funcionalidade	Notação gráfica	Entrada, Parâmetros e Saída
Sort	Ordenação dos dados		<p>Entrada: Unária – qualquer conjunto de dados.</p> <p>Parâmetros: Obrigatório – lista de atributos para ordenação e critério (asc ou desc) para cada atributo da lista.</p> <p>Saída: Unária – o mesmo conjunto de dados fornecido, com os dados ordenados</p>
Update	Atualização do valor de algum atributo		<p>Entrada: Unária – qualquer conjunto de dados.</p> <p>Parâmetros: Obrigatório – lista de atribuições.</p> <p>Saída: Unária – mesmo conjunto de dados fornecido, com valores alterados</p>
Copy	Criação de uma réplica dos dados		<p>Entrada: Unária – qualquer conjunto de dados.</p> <p>Saída: Binária – o próprio conjunto de dados fornecido e uma réplica.</p>

Operadores de Agregação			
Operadores	Funcionalidade	Notação Gráfica	Entrada, Parâmetros e Saída
Sum	Soma os valores de atributos determinados		<p>Entrada: Unária – qualquer conjunto de dados.</p> <p>Parâmetros: (i) Obrigatório – lista de atributos, (ii) condição para a soma.</p> <p>Saída: Unária – conjunto de dados com a soma dos valores dos atributos.</p>
SumGroup	Soma os valores de atributos determinados para cada grupo		<p>Entrada: Unária – qualquer conjunto de dados.</p> <p>Parâmetros: (i) Obrigatório – lista de atributos, (ii) lista de atributos para agrupamento.</p> <p>Saída: Unária – conjunto de dados com a soma dos valores dos atributos considerando o agrupamento.</p>
Count	Quantifica os diferentes valores dos atributos determinados		<p>Entrada: Unária – qualquer conjunto de dados.</p> <p>Parâmetros: (i) Obrigatório – lista de atributos para quantificação, (ii) condição para a contagem.</p> <p>Saída: Unária – conjunto de dados com as quantidades de valores dos atributos.</p>
CountGroup	Quantifica os diferentes valores dos atributos determinados para cada grupo		<p>Entrada: Unária – qualquer conjunto de dados.</p> <p>Parâmetros: (i) Obrigatório – lista de atributos para quantificação, (ii) condição para a contagem, (iii) Obrigatório - lista de atributos para agrupamento.</p> <p>Saída: Unária – conjunto de dados com as quantidades de valores dos atributos considerando o agrupamento.</p>
Min	Determinação do menor valor dos atributos determinados		<p>Entrada: Unária – qualquer conjunto de dados.</p> <p>Parâmetros: (i) Obrigatório – lista de atributos para cálculo, (ii) condição para o cálculo.</p> <p>Saída: Unária – conjunto de dados com o menor valor de cada atributo.</p>

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

Operadores de Inicialização			
Operadores	Funcionalidade	Notação gráfica	Entrada, Parâmetros e Saída
SetNullAsDefault	Inicialização de atributos com valor nulo		<p>Entrada: Unária – qualquer conjunto de dados.</p> <p>Parâmetros: (i) Obrigatório – atributo para inicialização, (ii) Opcional – uma condição de seleção de itens para inicialização.</p> <p>Saída: Unária – o mesmo conjunto de dados com o atributo inicializado</p>
SetDefaultValue	Inicialização de atributos com valores específicos		<p>Entrada: Unária – qualquer conjunto de dados.</p> <p>Parâmetros: (i) Obrigatório – lista de atribuições, (ii) Opcional – uma condição de seleção de itens para inicialização.</p> <p>Saída: Unária – o mesmo conjunto de dados com os atributos inicializados.</p>
SurrogateKey	Inicialização de chave artificial		<p>Entrada: Unária – qualquer conjunto de dados.</p> <p>Parâmetros: Obrigatório – atributo chave.</p> <p>Saída: Unária – o mesmo conjunto de dados com a chave populada</p>
Sequence	Inicialização de atributo com valor sequencial		<p>Entrada: Unária – qualquer conjunto de dados.</p> <p>Parâmetros: Obrigatório – uma atribuição inicial.</p> <p>Saída: Unária – o mesmo conjunto de dados com o atributo populado com um valor sequencial</p>

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. 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. Saída: Unária – um único conjunto de dados
Sincronize	Sincronização de tarefas que executaram em paralelo		Entrada: Binária ou n-ária – tarefas que executarão em paralelo. Parâmetros: Obrigatório – uma condição de espera para cada entrada (todas devem ser cumpridas). Saída: Unária – um único conjunto de dados
Delay	Espera para a execução de uma atividade		Entrada: Unária – um conjunto de dados qualquer Parâmetros: Obrigatório – uma condição de espera (que deve ser cumprida). 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. 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 especificidade		<p>Entrada: N-ária – conjuntos de dados que serão processados.</p> <p>Parâmetros: (i) Obrigatório – título da função, (ii) Opcional – catálogo descritivo – detalhamento da rotina ou função.</p> <p>Saída: Comumente unária, mas pode ser n-ária – conjuntos de dados resultantes da aplicação da função ou rotina.</p>
SubFlow	Indicação do agrupamento de tarefas de ETL compondo um subfluxo		<p>Entrada: Comumente unária, mas pode ser n-ária – conjuntos de dados sobre os quais serão aplicadas tarefas de transformação, limpeza ou agregação.</p> <p>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.</p> <p>Saída: Comumente unária, mas pode ser n-ária – conjuntos de dados resultantes da aplicação de tarefas de transformação, limpeza ou agregação</p>

5.4 Considerações finais

Nesse capítulo está descrita a proposta de modelagem conceitual para processos de ETL baseada em operadores, o Intuitive, com os elementos gráficos usados para representar uma abstração das tarefas do processo. Foram propostas seis categorias de operadores: operadores de armazenamento, manipulação de dados, de agregação, de inicialização, de fluxo e de operadores especiais. As avaliações aplicadas à essa abordagem e os resultados obtidos são descritos a seguir.