Análise de Dados com Base em Processamento Massivo em Paralelo

Processo de ETL/ELT

Profa. Dra. Cristina Dutra de Aguiar Ciferri

Resumo:

O processo de ETL/ELT se refere à extração (E), à transformação (T) e à carga (L) dos dados no *data warehouse*, no *data mart* ou no *data lake*. A transformação engloba várias operações que atuam sobre os dados: tradução, limpeza e integração. O processo de ETL/ELT representa a atividade mais complexa, cara e demorada do ambiente de *data warehousing*, sendo essencial ao bom funcionamento do ambiente. Neste texto, são detalhadas as operações que compõem o processo de ETL/ELT.

Conteúdo

1 Contextualização **3**
1.1 Projeto do Data Warehousing . . . 3
1.2 Base das Explicações . . . 4
1.3 Processo de ELT e Big Data . . . 4
1.4 Diferença entre Instância e Esquema . . . 5

2 Operações do Processo de ETL/ELT **6**
2.1 Extração . . . 6
2.1.1 Exemplos de Abordagens de Extração . . . 7
2.1.2 Detecção e Propagação de Alterações . . . 8
2.2 Tradução . . . 10
2.3 Limpeza . . . 11
2.4 Integração . . . 12
2.4.1 Integração de Esquemas . . . 12
2.4.2 Integração de Instâncias . . . 13
2.5 Carga . . . 14

3 Exemplo do Processo de ETL **15**
3.1 Projeto da Aplicação de Data Warehousing . . . 15
3.2 Extração . . . 17
3.3 Tradução . . . 18
3.4 Limpeza . . . 18
3.5 Integração . . . 18
3.6 Carga . . . 20

4 Conclusão **20**

1 CONTEXTUALIZAÇÃO

O processo de ETL/ELT se refere à extração (E), à transformação (T) e à carga (L) dos dados em um local de armazenamento dos dados presentes no *data warehousing*. A transformação engloba várias operações que atuam sobre os dados: tradução, limpeza e integração. Quando se tem o processo de ETL, os dados das fontes de dados são extraídos, transformados e carregados no *data warehouse* (DW) e nos *data marts*. Quando se tem o processo de ELT, os dados das fontes de dados são extraídos, armazenados no *data lake* e depois transformados para serem carregados no DW e nos *data marts*. Em ambos os processos, é possível usar a *data staging area* como uma área de armazenamento intermediária que contém os dados que vão sofrendo várias transformações até que estejam prontos para serem carregados no DW e nos *data marts*.

1.1 PROJETO DO DATA WAREHOUSING

Antes do processo de ETL/ELT ser iniciado, deve ser desenvolvido um projeto do *data warehousing*, para se determinar a melhor forma de migrar os dados das fontes de dados para o DW. Esse projeto deve ponderar diversos fatores, tais como o objetivo da aplicação de *data warehousing* a ser desenvolvida, os recursos disponíveis, *hardware* e *software* apropriados, as pessoas envolvidas e o planejamento da capacidade do ambiente.

Dentre as principais atividades envolvidas, pode-se citar:

- Identificar o propósito da aplicação de *data warehousing* e o volume de dados manipulado.
- Identificar a arquitetura de *data warehousing* que melhor se adéque aos propósitos e ao volume de dados, bem como definir os componentes da arquitetura.
- Instanciar a arquitetura, por meio da integração de servidores, ferramentas e tecnologias.
- Realizar o projeto do DW, dos *data marts* e do *data lake*, identificando os dados a serem armazenados, a organização física e a especificação de índices, dentre outros.
- Identificar as fontes de dados que possuem os dados relevantes para a aplicação, bem como os dados de interesse de cada fonte, usando como base o projeto dos locais de armazenamento de dados.
- Integrar as fontes de dados ao *data warehousing*.
- Definir políticas de acesso aos dados, incluindo aspectos de segurança.

Um importante aspecto a ser destacado é a Lei Geral de Proteção dos Dados (LGPD). Essa lei introduz mudanças jurídicas voltadas à proteção de dados pessoais dos brasileiros, como

nome, endereço, idade, estado civil, endereço de e-mail e situação patrimonial [3]. A lei estabelece a responsabilidade civil dos responsáveis pelo tratamento dos dados. Essa lei tem alto impacto no *data warehousing*, desde que o ambiente como um todo diz respeito a *dados*. São manipulados dados das fontes, sobre os quais aplica-se o processo de ETL/ELT, dados do *data lake*, que já podem ser consultados por ferramentas de análise e consulta, dados do DW e dos *data marts*.

Somente depois do projeto do *data warehousing* estar bem e especificado, deve ser iniciado o processo de ETL/ELT propriamente dito.

1.2 BASE DAS EXPLICAÇÕES

Neste texto, são detalhadas as operações que compõem o processo de ETL/ELT considerando como base de explicação a carga dos dados no DW. Entretanto, o detalhamento das operações se aplica também aos *data marts* e ao *data lake*, respeitando-se as particularidades de cada um desses locais de armazenamento. Somente quando algum comentário pertinente for necessário, são destacados aspectos específicos relacionados ao *data lake*. Em contrapartida, não são descritos aspectos específicos aos *data marts*, desde que cada *data mart* é um pequeno DW.

Adicionalmente, neste texto é detalhado o processo de ETL/ELT como um todo. Entretanto, existe uma abordagem alternativa na qual projeta-se apenas o processo de EL para a carga dos dados no *data lake*. Nesse caso, os dados são explorados no próprio *data lake* até que seja encontrado um conjunto de dados interessantes para serem analisados para a tomada de decisão estratégica. Quando isso acontece, projetam-se as operações do processo de TL para a transformação dos dados e a carga destes no DW.

Outro aspecto importante refere-se ao fato de que o processo de ETL/ELT é detalhado sem considerar o volume de dados dentro do contexto de *big data*. Isto está relacionado ao fato de que os princípios das operações envolvidas no processo são os mesmos, considerando-se, entretanto, as características específicas de *big data*.

1.3 PROCESSO DE ELT E BIG DATA

Apesar dos princípios das operações do processo de ELT serem os mesmos do que os princípios das operações do processo ETL, é importante destacar que o processo de ELT de gigantescos volumes de dados introduz grande complexidade adicional quando comparado com o processo considerando volumes de dados tradicionais.

Essa complexidade está relacionada com o modelo de Vs que define o conceito de *big data* [4, 5, 10, 13]. Ou seja, a complexidade advém da manipulação de um gigantesco volume de dados gerado em um pequeno intervalo de tempo, sendo que esses dados são de qualquer tipo, possuem diferentes graus de confiabilidade, podem incluir dados importantes ou dados desnecessários para a aplicação, sofrem alterações constantemente e precisam ser visualizados

adequadamente.

Por exemplo, em um processo de ELT de *big data* [5, 8]:

- A quantidade de fontes de dados, mesmo que referente a um domínio muito pequeno, é muito maior.
- A variedade de domínios é muito maior.
- Muitas fontes de dados são dinâmicas, indicando que uma gigantesca quantidade de dados torna-se continuamente disponível e pode ser usada para propósitos de análise em tempo (quase) real. Essa complexidade, em particular, refere-se à manipulação de dados em fluxo contínuo (*data streaming*).
- As fontes de dados são extremamente heterogêneas com relação ao formato dos dados, os quais podem ser estruturados, semiestruturados e não estruturados.
- Os dados apresentam grande variabilidade, dificultando ou até mesmo tornando impossível a identificação de mesmas entidades do mundo real presentes em diferentes fontes.
- Os dados das fontes de dados apresentam muita variação de qualidade, com diferenças significativas na acurácia e na cobertura.
- O tratamento incipiente do aspecto temporal é muito mais emergente.

Como resultado, todas as operações do processo são muito mais complexas, caras e demoradas.

1.4 DIFERENÇA ENTRE INSTÂNCIA E ESQUEMA

As operações do processo de ETL/ELT dependem da definição do *esquema* do DW. A partir dessa definição, pode-se decidir quais *dados* devem ser extraídos de quais fontes de dados, como os dados devem ser traduzidos do esquema de cada fonte de dados para o esquema do DW, como deve ser feita a limpeza e a integração dos dados e como os dados devem ser armazenados no DW. É necessário entender, portanto, a diferença entre os conceitos de esquema e de instância.

Instância se refere à coleção de dados que são armazenados em um banco de dados em um determinado momento. Ou seja, instância se refere ao *dado* propriamente dito. Exemplos de instância são "João", "José", "Maria", "R$1.500", "R$3.000", "R$5.000". Sinônimos de instância são extensão do banco de dados, linhas (ou tuplas) de tabelas relacionais e registros de arquivos.

Esquema se refere ao projeto do banco de dados, ou seja, aos metadados que descrevem os dados. Por exemplo, o metadado referente a "João", "José", "Maria" é nome do funcionário e o metadado referente a "R$1.500", "R$3.000", "R$5.000" é salário do funcionário. O esquema

inclui as entidades e os relacionamentos existentes entre essas. No exemplo, a entidade "funcionário" relaciona-se com a entidade "salário" por meio de um relacionamento "recebe". Sinônimos de esquema são intenção do banco de dados, definição de tabelas relacionais e definição da estrutura (campos) dos registros de arquivos.

Com relação ao *data lake*, desde que ele armazena dados estruturados, semiestruturados e não estruturados, costuma-se caracterizar que ele armazena dados em seu formato nativo. Entretanto, na prática, quando se tem dados tabulares ou aninhados, é usualmente comum transformá-los em um formato mais adequado para serem armazenados e explorados no *data lake*. Portanto, pode-se considerar também um projeto de esquema de *data lake*.

2 OPERAÇÕES DO PROCESSO DE ETL/ELT

As operações do processo de ETL/ELT são: (i) extração, incluindo extração inicial e extração incremental; (ii) transformação, incluindo tradução, limpeza e integração; e (iii) carga. Essas operações são descritas a seguir.

2.1 EXTRAÇÃO

A primeira operação importante do processo de ETL/ELT é a extração de dados de interesse a partir das fontes de dados. As tarefas da extração referem-se a: (i) quais dados são extraídos de quais fontes de dados; (ii) como esses dados são extraídos; (iii) com qual frequência esses dados devem ser periodicamente extraídos; e (iv) qual técnica empregar para identificar dados das fontes que foram alterados.

Para a carga inicial dos dados no DW, todos os dados de interesse das fontes de dados são extraídos e encaminhados para as demais operações do processo de ETL/ELT. Quanto maior for o volume de dados, mais tempo é consumido para a execução da operação de **extração inicial**.

Após a carga inicial, as fontes de dados podem continuar a alterar os seus dados. Visando a manutenção da consistência dos dados do DW com relação aos dados das fontes, essas alterações devem ser propagadas e incorporadas ao DW. Existem duas técnicas para se atualizar o DW: recomputação e atualização incremental, conforme detalhado a seguir.

- Recomputação. Nessa técnica, o conteúdo do DW é descartado e os seus dados são novamente carregados a partir das fontes de dados operacionais.
- Atualização incremental. Nessa técnica, apenas as alterações nos dados das fontes são propagadas ao *data warehousing*, as quais, juntamente com os dados armazenados no DW, são utilizadas para determinar o novo conteúdo do DW.

Técnicas de atualização incremental baseiam-se na heurística de que somente uma parte do DW é alterado em resposta às alterações realizadas nos dados de interesse das fontes de

[7]. Uma heurística tende a funcionar bem na grande maioria dos casos e, por isso, a atualização incremental é muito mais usada para a realização da extração periódica dos dados quando comparada com as técnicas de recomputação. Portanto, para as demais cargas realizadas usualmente ocorre uma **extração incremental**.

Dois aspectos importantes quanto à autonomia das fontes de dados devem ser considerados na extração: intervenção mínima e operação não intrusiva. O primeiro deles refere-se ao fato de que os sistemas que alimentam as fontes de dados não devem sofrer intervenções ou as intervenções devem ser mínimas.

O segundo aspecto diz respeito à execução da extração, a qual não pode impactar negativamente na execução dos demais sistemas envolvidos, ou seja, deve-se ter a característica de ser uma operação não intrusiva. Por isso, a operação de extração deve ser executada, sempre que possível, durante a "janela de manutenção", período no qual os sistemas operacionais envolvidos em geral ficam mais ociosos. Essa extração é usualmente conhecida como *extração por lote*. Isso significa que ela tem uma certa periodicidade (como exemplo diária, semanal ou mensal) e que todos os dados alterados de acordo com essa periodicidade são extraídos. Uma alternativa à janela de manutenção é fazer uma *extração por streaming*, ou seja, extrair os dados de maneira contínua em tempo (quase) real. Devido ao seu aspecto incremental de extração, o impacto na competição por recursos com a aplicação geradora de dados tende a ser mínimo. Como desvantagem, a extração por *streaming* introduz complexidade adicional, desde que os dados precisam ser continuamente extraídos. Adicionalmente, o *data lake* deve ser obrigatoriamente um dos componentes do *data warehousing*, sendo responsável por armazenar esses dados continuamente extraídos. Portanto, usualmente é mais fácil manter uma extração por lotes.

2.1.1 EXEMPLOS DE ABORDAGENS DE EXTRAÇÃO

A operação de extração pode ser implementada considerando diferentes abordagens, como exemplo: por meio de uma interface de comandos padronizados (interface comum), por meio de um protocolo comum de acesso aos dados e por meio de um conversor de comandos de manipulação de dados e de formatos de dados (*gateway*). Cada uma dessas três abordagens enfatiza um elemento distinto que é comum entre o cliente e o servidor. Considerando-se a operação de extração, uma aplicação que realiza a extração dos dados pode ser considerada um cliente, ao passo que cada fonte de dados é um servidor de dados.

A primeira abordagem usa a interface (tanto do cliente quanto do servidor) como um elemento comum para o acesso aos dados gerenciados por diferentes produtos. O princípio por trás desta abordagem consiste na especificação de programas de aplicação clientes de acordo com uma interface genérica (API - *Application Programming Interface*), a qual é independente do tipo do servidor. Em tempo de execução, a aplicação identifica qual a fonte de dados a ser acessada e um módulo específico (*driver*) converte os formatos de dados e os comandos pa-

dronizados para os formatos compreendidos pelo servidor alvo. Como resultado, o código da aplicação cliente não precisa ser alterado quando esta for redirecionada para outro tipo de servidor. Exemplos de padrões de interface comum incluem o ODBC (*Open Database Connectivity*) e o BDE (*Borland Database Engine*). Outro exemplo é o JDBC (Java Database Connectivity), o qual tem sido amplamente utilizado em ambientes computacionais paralela e distribuída. Por exemplo, o Apache Spark oferece funcionalidades para extrair dados de qualquer fonte que ofereça suporte ao JDBC.

A abordagem de protocolo comum utiliza um protocolo bem definido para conectar aplicações clientes aos vários tipos de servidores. Intuitivamente, cada interface cliente recebe requisições de aplicações clientes e as traduz em um protocolo comum apropriado. De forma similar, cada interface servidora identifica requisições geradas de acordo este protocolo, processa tais requisições e traduz os resultados obtidos de acordo com o protocolo comum. Essa abordagem permite que a interoperabilidade entre os tipos de servidores que utilizam o mesmo protocolo para a comunicação seja garantida mesmo que diferentes APIs tenham sido empregadas. Os padrões DRDA (*Distributed Relational Database Architecture*) e RDA (*Remote Data Access*) e REST (*Representational state transfer*) são exemplos de protocolos.

A abordagem de *gateway* é representada pelo uso de tradutores de comandos de acesso a dados (*gateways* para bancos de dados) como elementos comuns entre clientes e servidores. Esses tradutores assumem a funcionalidade de mapeamento de dados e de comandos entre os vários clientes e servidores. Por não serem padronizados, *gateways* reduzem a portabilidade das aplicações desenvolvidas, uma vez que estas utilizam as APIs por eles disponibilizadas. Como exemplos de produtos de *gateway* pode-se citar o Database Gateway para DB2 e o Informix Enterprise Gateway.

As abordagens podem ser utilizadas separadamente ou conjuntamente, de acordo com as necessidades da organização. A abordagem de *gateway* usualmente complementa ou estende as outras duas abordagens. Quando desenvolvida com alguma combinação de interface comum ou protocolo comum, adiciona flexibilidade à arquitetura do sistema.

2.1.2 DETECÇÃO E PROPAGAÇÃO DE ALTERAÇÕES

A extração incremental dos dados inicia-se com a detecção e a propagação de alterações nos dados das fontes que participam do *data warehousing*. Rotinas de detecção de alterações devem (i) monitorar as modificações ocorridas nas fontes de dados (ou seja, inserções, atualizações e remoções de dados); (ii) identificar as diferenças entre o estado corrente das fontes de dados e o estado que havia na extração anterior; e (iii) extrair somente os dados necessários, diminuindo assim o volume de dados manipulado e provendo uma redução significativa no tempo necessário para a execução da extração.

A periodicidade da incorporação das alterações aos dados do DW depende das necessidades de análise dos usuários de suporte à decisão e do nível de consistência desejado. Siste-

mas comerciais usualmente atualizam seus dados com periodicidade diária ou semanal. No entanto, caso consultas OLAP (*on-line analytical processing*) requeiram dados correntes, cada simples alteração realizada nas fontes de dados deve ser propagada continuamente. A frequência de incorporação das alterações é determinada pelo administrador do DW, podendo ser diferente para fontes de dados distintas.

A detecção de alterações nos dados operacionais é realizada de acordo com as facilidades oferecidas pelas fontes de dados. Assim, rotinas de detecção de alterações devem considerar como a fonte é capaz de indicar que seus dados foram alterados. As técnicas empregadas na implementação dessas rotinas são denominadas *Change Data Capture* (CDC) [11, 9]. Essas técnicas podem ser baseadas em:

- *Timestamp* (marcadores de tempo). Existem fontes de dados que marcam os dados alterados com um *timestamp* em uma coluna de auditoria. O CDC compara essa informação com a data e o horário da extração de dados mais recente. Os dados que têm a data de alteração maior do que a data da extração anterior são extraídos para serem utilizados na atualização do DW. Como exemplo, pode-se citar o software Kafka Connect[1], da empresa Confluent, que realiza esse tipo de extração e envia os dados para o Apache Kafka[2].

- *Triggers* (gatilhos). O recurso de gatilhos presente nos sistemas gerenciadores de banco de dados (SGBDs) relacionais pode ser usado para disparar o CDC tanto para a detecção quanto para a notificação automática de alterações que ocorrem nos dados da fonte de dados. O comando CREATE TRIGGER encontra-se presente em vários SGBDs, como exemplo Oracle, PostgreSQL e DB2.

- *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 das fontes de dados para identificar as diferenças que devem ser extraídas para atualização do DW. Como exemplo, pode-se citar o software Logstash[3], da empresa Elastic.

- Comparação de *snapshots* (instantâneos). Um *snapshot* é como uma foto dos valores de dados armazenados em um certo momento no banco de dados, ou seja, uma foto do estado do banco de dados. Nesse caso, a operação 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 de dados, gerando um novo estado que é, então, capturado quando a operação 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

[1] https://docs.confluent.io/current/connect/index.html
[2] https://kafka.apache.org/intro
[3] https://www.elastic.co/pt/logstash

fonte de dados e o estado da fonte de dados na extração anterior. As diferenças identificadas resultam em um arquivo delta, o qual é então usado para a atualização incremental do DW.

As técnicas possuem limitações e desvantagens, o que faz com que o processo de detecção de alterações nos dados das fontes de dados represente um grande desafio para o *data warehousing*. Por exemplo, somente poucos dados operacionais possuem *timestamp* a ele associados. Adicionalmente, o uso da técnica de *timestamp* não permite identificar dados que tenham sido removidos da fonte de dados. Similarmente, somente poucas fontes de dados oferecem recursos de gatilhos. Ainda com relação ao uso de gatilhos, destaca-se que essa técnica pode demandar intervenções nos sistemas que alimentam as fontes de dados, além da possibilidade de onerar a execução desses sistemas. Com relação ao uso do arquivo de *log*, muitas barreiras devem ser enfrentadas. Geralmente esse arquivo somente pode ser acessado utilizando-se o privilégio de administrador de banco de dados. Ademais, seu formato interno é construído para propósitos específicos do sistema, podendo ser difícil de ser entendido e conter informações adicionais desnecessárias. Ademais, o *log* é considerado um componente essencial no processo de recuperação de falhas, sendo protegido pelo sistema. Já o problema da técnica de comparação de *snapshots* está relacionado ao fato de que, à medida que o volume de dados da fonte de dados cresce, comparações cada vez maiores precisam ser realizadas.

A decisão sobre qual a melhor técnica de CDC para a extração de dados no processo de ETL deve considerar os sistemas que alimentam das fontes de dados e a estrutura desses dados (estruturados, semiestruturados ou não estruturados). Segundo [11], soluções de CDC que fazem a comparação de *snapshots* são as mais comumente usadas na operação de extração do processo de ETL em *data warehousing*.

2.2 TRADUÇÃO

A operação de tradução consiste na conversão dos dados do formato nativo das fontes de dados para o formato e o modelo utilizados pelo *data warehousing*. Esse processo, assim como o de extração dos dados, é altamente dependente da fonte sendo considerada. Por exemplo, se a fonte de dados armazena seus dados de acordo com o formato JSON (*JavaScript Object Notation*) e o DW utiliza o modelo relacional, rotinas de tradução dos dados do formato JSON para o modelo relacional devem ser especificadas. Uma vez que as fontes de dados que participam do *data warehousing* armazenam seus dados segundo diferentes formatos, diversas rotinas de tradução devem ser desenvolvidas. Essas traduções devem incluir tanto transformações de esquema (alteração da estrutura dos dados) quanto transformações dos dados correspondentes.

Um aspecto muito importante do carregamento dos dados refere-se à característica temporal dos dados do DW. A maioria das fontes de dados não é histórica, ou seja, elas armazenam somente valores correntes dos dados. Em contrapartida, o DW deve armazenar dados históricos, uma vez que os usuários de suporte à decisão usualmente estão interessados no histórico

de como os dados das fontes evoluíram ao longo do tempo. Desta forma, durante o processo de tradução dos dados, dados temporais devem ser associados aos dados sendo manipulados. Esses dados temporais podem refletir tanto o momento de atualização dos dados na fonte de dados quanto o momento de armazenamento desse no DW, ou ambos.

2.3 LIMPEZA

Desde que o *data warehousing* é utilizado para suporte à decisão, é importante que os dados mantidos no ambiente sejam corretos e de qualidade. Em outras palavras, contribuições potenciais dependem da qualidade das decisões tomadas, as quais, por sua vez, dependem da qualidade dos dados utilizados no processo decisório. Em especial, dentre os diversos fatores que podem acarretar o insucesso de uma aplicação de *data warehousing*, a falta de atenção com relação à qualidade dos dados representa um fator extremamente comprometedor. Dados duvidosos e/ou incorretos podem gerar informações errôneas que causam um efeito adverso nos lucros da corporação.

A qualidade dos dados está diretamente relacionada às regras de negócio a eles associadas. Uma vez que estas regras são definidas, o dado deve ser testado para verificar a sua consistência e a sua completude com relação a essas regras. Tais regras podem ser inferidas tanto por meio da identificação das regras de negócio da corporação e a posterior associação dessas regras aos dados quanto por meio da análise dos próprios dados e da identificação de padrões que se aplicam à maioria dos dados analisados. Se, por exemplo, em 95% dos registros analisados indicam que a idade das pessoas que ocupam cargos gerenciais e que ganham salários acima de R$40.000,00 é superior a 40 anos, um registro com valor de idade igual a 18 anos deve ser examinado cautelosamente.

Existem diversas situações nas quais os valores de dados armazenados podem estar incorretos e, consequentemente, nas quais a limpeza dos dados torna-se necessária: comprimentos de campos inválidos, dados incompletos ou em branco, duplicações, descrições inconsistentes, violação de restrições de integridade, associações de valores inconsistentes, abreviações de valores não padronizadas e utilização de caracteres ou de tipos de dados não desejados.

A atribuição do valor 350 para um campo idade de um funcionário é um exemplo de associação de valor inconsistente, ao passo que a existência de uma chave de funcionário em um arquivo que não corresponde às chaves do arquivo de funcionários representa uma violação da restrição de integridade. Outros exemplos incluem campos com valores de cor e telefone incorretos, bem como endereços parcialmente preenchidos. Por exemplo, somente o campo CEP é preenchido, sendo os demais campos, nome da rua, número, complemento, cidade, estado, país, deixados em branco. Ainda outro exemplo diz respeito às datas, as quais podem ser armazenadas erroneamente na forma ddmmaaaa com significado mmddaaaa. Por exemplo, a data 01032020 pode indicar 01 de março de 2020 ou 03 de janeiro de 2020.

Por ser uma atividade de extrema importância para o sucesso do *data warehousing*, o pro-

cesso de limpeza dos dados não deve ser realizado como uma atividade separada, mas sim durante todas as demais atividades de extração, de tradução, de integração e de carga dos dados no DW.

2.4 INTEGRAÇÃO

Assim como em qualquer ambiente que envolve várias fontes de dados heterogêneas, em um *data warehousing* existe uma alta probabilidade de se existir várias cópias do mesmo dado em diferentes fontes de dados (representadas de forma igual ou não) ou dados correlatos armazenados em diferentes fontes que são inconsistentes entre si. Essa diversidade de representações está relacionada, em primeiro lugar, ao fato de que as aplicações representadas por essas fontes de dados foram criadas independentemente, de forma não coordenada e sem considerar que um dia seriam integradas. Depois, a modelagem de cada aplicação varia de acordo com diversos fatores, tais como as necessidades, os objetivos e o universo de atuação de cada aplicação. Esses fatores determinam entidades próprias a cada domínio. Por fim, diferentes analistas podem possuir diferentes percepções do mundo real, originando, muitas vezes, modelos distintos para uma mesma aplicação.

Como resultado desta diversidade de representações, a operação de integração depende da identificação de similaridades e de diferenças existentes entre os dados das fontes de dados que foram previamente extraídos, traduzidos e limpos, além da identificação de conjuntos desses dados que, apesar de serem distintos entre si, são relacionados por alguma propriedade semântica [1, 2]. Estas similaridades e diferenças devem ser detectadas tanto em **nível de esquema** quanto em **nível de instância**.

2.4.1 INTEGRAÇÃO DE ESQUEMAS

A integração de esquemas refere-se à (i) criação de um esquema (global) mediador e (ii) identificação de mapeamentos entre o esquema (global) mediador e os esquemas de cada fonte de dados que fornece dados de interesse para a aplicação. No caso do ambiente de *data warehousing*, o esquema (global) mediador é o esquema do DW. Portanto, devem ser definidos mapeamentos que especificam como o esquema de cada fonte de dados se acomoda com relação ao esquema do DW.

Considerando-se o nível de esquema, os conflitos existentes entre os dados a serem integrados podem ser divididos em três grupos: conflitos de nome, conflitos semânticos e conflitos estruturais. O primeiro tipo de conflito, conflito de nome, refere-se aos nomes utilizados para representar os diferentes elementos existentes nos esquemas a serem integrados. Diferentes nomes podem ser aplicados ao mesmo elemento (problema dos sinônimos) ou o mesmo nome pode ser aplicado a diferentes elementos (problema dos homônimos). Um exemplo de sinônimo ocorre quando os nomes *funcionário* em um esquema, *colaborador* em outro esquema e

empregado em um terceiro esquema são utilizados para representar os funcionários. Um exemplo de homônimo ocorre quando o nome *data* é usado para representar a *data de contratação* do funcionário em um esquema e a *data de nascimento* do funcionário em outro esquema.

Após a identificação dos conflitos de nome, devem ser identificados os conflitos semânticos. Esse tipo de conflito surge quando o mesmo elemento é modelado nos diferentes esquemas, porém representando conjuntos que se sobrepõem. Em outras palavras, o conjunto de instâncias do elemento de um esquema é mais abrangente do que o conjunto de instâncias do elemento do outro esquema. Por exemplo, em uma fonte de dados, o elemento funcionário se refere aos funcionários da área de Engenharia, em outra fonte de dados, o elemento funcionário se refere aos funcionários da área de *Marketing* e, em uma terceira fonte de dados, o elemento funcionário representa os funcionários da área de Recursos Humanos. Os funcionários podem ser diferentes entre si. Adicionalmente, o mesmo funcionário pode estar presente em mais de uma fonte de dados, por exemplo quando ele muda de área de atuação durante a sua trajetória dentro da empresa. O funcionário começa a trabalhar na área de Recursos Humanos e depois muda para a área de *Marketing*.

Finalmente, conflitos estruturais surgem sempre que diferentes construtores estruturais são utilizados para modelar o mesmo conceito representado em diferentes aplicações. Como exemplo, considerando-se o modelo entidade-relacionamento, o mesmo conjunto de objetos do mundo real pode ser representado como um tipo-entidade em um esquema e como um atributo de um tipo-entidade em outro esquema. Ou seja, em um primeiro esquema, o endereço do funcionário é modelado como atributo de funcionário e, em um segundo esquema, endereço é modelado como um tipo-entidade. A diversidade de conflitos estruturais que podem aparecer em um problema de integração depende da semântica do modelo de dados utilizado.

Em geral, as discrepâncias existentes entre os esquemas apresentam mais do que um tipo de conflito. Adicionalmente, somente depois de realizada a integração de esquemas, inicia-se a integração de instâncias.

2.4.2 INTEGRAÇÃO DE INSTÂNCIAS

Em um ambiente de *data warehousing*, não somente a integração dos esquemas das diferentes aplicações é importante, mas também a integração das suas instâncias correspondentes. Existem dois tipos de integração de instâncias: ambiguidade na identificação de entidades e resolução de conflitos de valores de atributos.

A ambiguidade na identificação de entidades tem como objetivos: (i) identificar quais entidades das fontes de dados heterogêneas referem-se à mesma entidade do mundo real; e (ii) agrupar essas entidades em agrupamentos de entidades similares. Dois diferentes cenários podem ocorrer. No primeiro cenário, as entidades das fontes de dados heterogêneas podem ser identificadas univocamente por meio de chaves, as quais distinguem uma entidade da outra. Por exemplo, cada funcionário de uma empresa é identificado por um número funcional,

que é uma chave que distingue esse funcionário univocamente dentro da empresa. Nesse cenário, o agrupamento dos dados dos funcionários obtidos das fontes de dados é feito por meio do agrupamento de cada número funcional.

No segundo cenário, as entidades das fontes de dados heterogêneas não podem ser identificadas univocamente por meio de chaves. Esse cenário é muito comum, por exemplo, quando os dados são obtidos de páginas web. Mesmo que haja um código de um produto, o valor desse código varia de página web a página web para o mesmo produto. Nesse caso, a atividade de integração torna-se bem mais complexa, sendo necessária a aplicação de técnicas avançadas para a identificação o mais automática possível de entidades similares. Na literatura, essas técnicas têm sido chamadas de encadeamento de registros (*record linkage*), deduplicação, combinação de referências, identificação de objetos, incerteza de entidade, reconciliação de referências e resolução de entidades.

Depois de identificadas quais entidades das fontes de dados heterogêneas referem-se à mesma entidade do mundo real, deve ser realizada a resolução de conflitos de valores dos atributos dessas entidades. O objetivo da resolução de conflitos é resolver inconsistências nos valores dos dados das entidades que se referem à mesma entidade do mundo real, mas que diferem nos valores dos seus atributos.

Como um exemplo simples da necessidade de resolução de conflitos, os dados correspondentes ao atributo sexo do funcionário, o qual foi obtido de diferentes fontes de dados, podem ser codificados de forma inconsistente. Em uma primeira fonte de dados, esse campo é instanciado com os valores “M” (masculino) ou “F” (feminino). Em outra fonte, os valores permitidos são “0” (masculino) ou “1” (feminino). Em uma terceira fonte, os valores assumidos podem ser “H” (homem) ou “M”(mulher). Independentemente do formato final do campo integrado sexo a ser armazenado no DW, os diferentes valores devem ser corretamente decifrados antes de serem armazenados. Outro exemplo consiste em um mesmo campo largura armazenado em quatro diferentes fontes de dados, porém com valores que representam unidades de medidas distintas: centímetros, metros, polegadas e jardas. Ainda outro exemplo se refere à data de nascimento. Em uma primeira fonte de dados, ela é representada como 1990-02-02, enquanto que em uma segunda fonte de dados, ela é representada como 02/02/1995.

Como resultado do processo de integração, tem-se uma visão integrada dos dados do DW, sem valores repetidos ou inconsistências, e de alta qualidade. Portanto, os dados podem ser carregados no DW.

2.5 CARGA

Após as operações de extração, de tradução, de limpeza e de integração, ocorre o armazenamento dos dados pré-processados no DW. Durante essa operação, vários processamentos adicionais ainda são realizados. Por exemplo:

- Geração de agregações (ou visões materializadas), que são armazenadas adicionalmente

aos dados base do DW visando a melhoria no desempenho do processamento das consultas OLAP.

- Necessidade de construção de índices frente às análises a serem realizadas. Por um lado, índices melhoram o desempenho no processamento de consultas e são amplamente utilizados em aplicações de banco de dados. Por outro lado, eles introduzem sobrecarga adicional devido à necessidade de manutenção. Neste sentido, usualmente todos os índices são desabilitados antes do processo de carga e habilitados após o processo de carga. A habilitação do índice permite que o mesmo seja reconstruído, refletindo portanto no novo estado do DW após a carga.

- A verificação de restrições de integridade deve ser considerada com cuidado. Assim como para os índices, usualmente desabilita-se a verificação dessas restrições antes da carga, para depois habilitá-las novamente. Isso está relacionado ao fato de que a verificação de restrições de integridade é uma funcionalidade computacionalmente cara, principalmente para o carregamento de grandes volumes de dados. Em especial, o processo de limpeza realizado nos dados deve garantir as restrições de integridade referencial. Como resultado, a integridade referencial dos dados do DW é garantida mesmo com a verificação desativada.

- Necessidade de ordenação dos dados quando apropriado.

3 EXEMPLO DO PROCESSO DE ETL

Considere a empresa **BI Solutions** usada como exemplo ao longo da disciplina. Considere uma aplicação de *data warehousing* da empresa voltada à folha de pagamento. O propósito da aplicação consiste em investigar os gastos em salários dos funcionários e a quantidade de lançamentos na folha de pagamento. Para tanto, devem ser considerados dados relativos aos funcionários, cargos ocupados por estes, filiais nas quais eles trabalham e datas nas quais recebem pagamento. Para simplificar as atividades do processo de ETL, nos exemplos descritos nesta seção é considerada apenas a parte da aplicação referente aos **funcionários**.

3.1 PROJETO DA APLICAÇÃO DE DATA WAREHOUSING

Na Figura 1 é ilustrada a arquitetura que melhor se adéqua aos propósitos da aplicação e ao volume de dados manipulados. A arquitetura é baseada no processamento de dados em lote, desde que pretende-se realizar a extração mensal dos dados das fontes. Adicionalmente, o volume de dados considerado é muito pequeno, desde que o exemplo tem como objetivo principal facilitar o entendimento das atividades do processo de ETL/ELT. Com um pequeno volume de

dados, é possível compreender melhor as correspondências existentes. Como resultado, o processo de ETL é executado somente por meio do Pandas[4], sendo o resultado armazenado em um *data mart* que deve ser gerenciado por meio do relacional MySQL[5]. Pretende-se realizar as análises usando a linguagem de programação Python. Caso o volume de dados fosse muito maior, a arquitetura poderia incorporar uma *data staging area* para armazenamento intermediário. A arquitetura também pode, posteriormente, incorporar um DW.

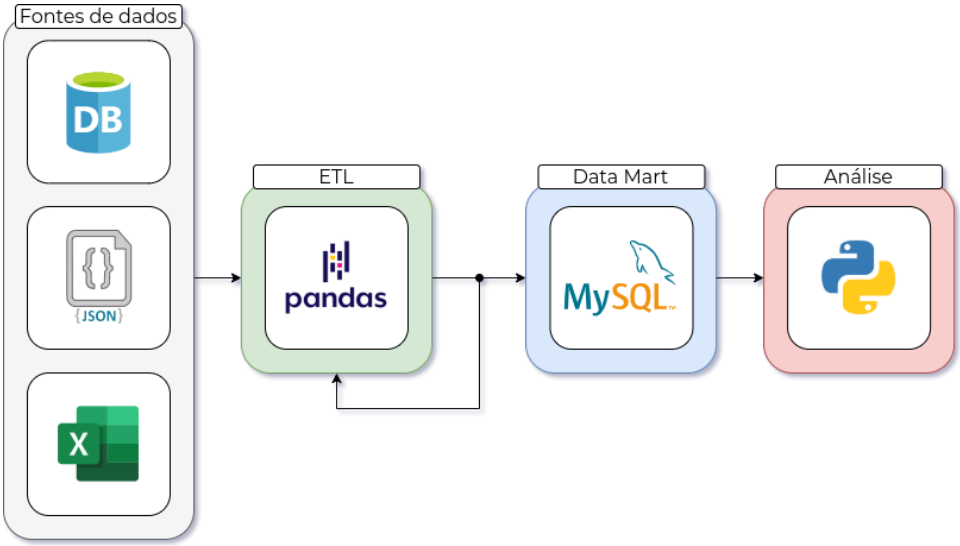

Figura 1: *Pipeline* da aplicação exemplo.

O *data mart* é baseado no modelo relacional. De acordo com modelo relacional, o banco de dados é representado por um conjunto de esquemas de relação, sendo que cada esquema de relação possui um nome único e é uma tabela bidimensional [6]. Cada coluna da tabela tem um nome distinto e representa um atributo. Cada atributo possui um domínio, que é o conjunto de valores de dados que aquele atributo pode assumir, de acordo com um tipo de dado. Cada linha da tabela representa o relacionamento entre um conjunto de valores, ou seja, representa uma tupla (ou instância) do esquema de relação. Existem várias restrições de integridade que podem ser definidas sobre um esquema de relação. Uma delas é a restrição de chave primária, a qual possibilita que cada tupla seja identificada univocamente.

O projeto do *data mart* refere-se ao esquema de relação *funcionario*. Esse esquema é definido a seguir, sendo que a chave primária é representada pelo atributo funcPK. O conjunto de valores que esse atributo pode assumir é gerado artificialmente e consiste de números inteiros em ordem crescente. Os nomes dos demais atributos foram definidos de forma a terem significado semântico.

[4]https://pandas.pydata.org/
[5]https://www.mysql.com/

funcionario (funcPK, funcMatricula, funcNome, funcSexo, funcDataNascimento, funcDiaNascimento, funcMesNascimento, funcAnoNascimento, funcCidade, funcEstadoNome, funcEstadoSigla, funcRegiaoNome, funcRegiaoSigla, funcPaisNome, funcPaisSigla)

Usando como base o projeto do *data mart*, foram identificadas três fontes de dados que possuem os dados relevantes para a aplicação: (i) *funcionarioRelacional*, um banco de dados relacional; (ii) *colaboradorJSON*, um arquivo JSON, representando uma base de dados NoSQL (Not only SQL); e (iii) *empregadoPlanilha*, uma planilha Excel.

3.2 EXTRAÇÃO

As tarefas da extração referem-se a: (i) quais dados são extraídos de quais fontes de dados; (ii) como esses dados são extraídos; (iii) com qual frequência esses dados devem ser periodicamente extraídos; e (iv) qual técnica empregar para identificar dados das fontes que foram alterados. Nesta seção são descritas as decisões tomadas para o exemplo corrente considerando cada uma dessas tarefas.

Para a fonte *funcionarioRelacional*, devem ser extraídos os seguintes dados de interesse.

funcionarioRelacional (funcMatricula, funcNome, funcSexo, funcDataNasc, funcCidade, funcEstado, funcPais)

Para a fonte *colaboradorJSON*, devem ser extraídos os seguintes dados de interesse.

```
[{"colab_matricula": " ", "colab_nome": " ", "colab_sexo": " ", "colab_data_nasc": " ",
  "colab_cidade": " ", "colab_estado": " ", "colab_pais": " "}, ...]
```

Para a fonte *empregadoPlanilha*, devem ser extraídos os seguintes dados de interesse.

	A	B	C	D	E	F
1	Matrícula do Empregado	Nome do Empregado	Sexo do Empregado	Data de Nascimento	Cidade de Residência	Estado de Residência

Figura 2: Fonte de dados *empregadoPlanilha*.

Os dados de interesse devem ser extraídos por meio de três APIs, cada uma específica para uma fonte de dados. O objetivo da aplicação é realizar o processamento em lote, sendo a extração incremental feita com periodicidade mensal. Por fim, a técnica usada para a extração incremental da fonte *funcionarioRelacional* é o uso de *triggers*, desde que essa fonte é um SGBD relacional. Para as demais fontes, *colaboradorJSON* e *empregadoPlanilha*, deve ser usada a técnica de comparação de *snapshots*. Essas duas últimas fontes representam arquivos que não possibilitam o uso de outras técnicas para identificar as mudanças ocorridas nos dados.

3.3 TRADUÇÃO

Três diferentes rotinas de tradução de esquemas devem ser desenvolvidas, conforme descrito a seguir:

- *RotinaRelacionalRelacional*: para a tradução do esquema de *funcionarioRelacional* para o esquema relacional de *funcionario*.
- *RotinaJSONRelacional*: para a tradução do esquema *colaboradorJSON* para o esquema relacional de *funcionario*.
- *RotinaExcelRelacional*: para a tradução do esquema *empregadoPlanilha* para o esquema relacional de *funcionario*.

A operação de tradução é misturada com as operações de integração e de limpeza. Portanto, mais detalhes sobre as rotinas de tradução com relação aos esquemas e também com relação aos dados são apresentados posteriormente.

3.4 LIMPEZA

Não são descritos detalhes específicos relacionados à limpeza dos dados, desde que esse assunto foi amplamente discutido em outra disciplina do MBA em Ciência de Dados. Por hora, destaca-se que foi feito o uso de estratégias aprendidas em outras disciplinas do curso, com destaque para "Técnicas Avançadas de Captura e Tratamento de Dados". Exemplos de estratégias incluem manipulação de valores nulos, redundância e desbalanceamento de dados, detecção de *outliers* e tratamento de informações faltantes e errôneas.

3.5 INTEGRAÇÃO

A primeira etapa da operação de integração é a integração de esquemas. Para tanto, são definidos mapeamentos que especificam como o esquema de cada fonte de dados *funcionarioRelacional*, *colaboradorJSON* e *empregadoPlanilha* se relaciona com o esquema do DW.

Os mapeamentos definidos para o exemplo corrente utilizam como base a nomenclatura de correspondências entre os elementos dos esquemas [12]. Nessa nomenclatura, o símbolo $\equiv$ representa correspondência entre esquemas de relação e o símbolo $=$ representa correspondência entre atributos.

As correspondências entre os esquemas de *funcionario* e *funcionarioRelacional* são:

1. *funcionario* ≡ *funcionarioRelacional*
   (a) funcMatricula = funcMatricula
   (b) funcNome = funcNome
   (c) funcSexo = funcSexo
   (d) funcDataNascimento = funcDataNasc
   (e) funcCidade = funcCidade
   (f) funcEstadoSigla = funcEstado
   (g) funcPaisNome = funcPais

As correspondências entre os esquemas de *funcionario* e *colaboradorJSON* são:

2. *funcionario* ≡ *colaboradorJSON*
   (a) funcMatricula = colab_matricula
   (b) funcNome = colab_nome
   (c) funcSexo = colab_sexo
   (d) funcDataNascimento = colab_data_nasc
   (e) funcCidade = colab_cidade
   (f) funcEstadoSigla = colab_estado
   (g) funcPaisSigla = colab_pais

As correspondências entre os esquemas de *funcionario* e *empregadoPlanilha* são:

3. *funcionario* ≡ *empregadoPlanilha*
   (a) funcMatricula = Matrícula do Empregado
   (b) funcNome = Nome do Empregado
   (c) funcSexo = Sexo do Empregado
   (d) funcDataNascimento = Data de Nascimento
   (e) funcCidade = Cidade de Residência
   (f) funcEstadoNome = Estado de Residência

Os nomes *funcionario*, *funcionarioRelacional*, *colaboradorJSON* e *empregadoPlanilha* são sinônimos, ou seja, possuem conflito de nome. Adicionalmente, os funcionários presentes nos esquemas de relação representados por esses nomes consistem de conjuntos de funcionários

que se sobrepõem. Portanto, *funcionario*, *funcionarioRelacional*, *colaboradorJSON* e *empregadoPlanilha* também possuem conflito semântico. Com relação aos atributos, existem vários problemas de conflito de nome, a saber: (i) funcDataNascimento = funcDataNasc e funEstadoSigla = funcEstado, para as correspondências entre *funcionario* e *funcionarioRelacional*: ; (ii) todas as correspondências de atributos entre *funcionario* e *colaboradorJSON*; e (ii) todas as correspondências de atributos entre *funcionario* e *empregadoPlanilha*.

Depois de realizada a integração de esquemas, pode ser conduzida a integração de instâncias. Com relação à ambiguidade na identificação de entidades, foi considerado o fato de que as entidades das fontes de dados *funcionarioRelacional*, *colaboradorJSON* e *empregadoPlanilha* podem ser identificadas univocamente por meio de chaves. Ou seja, cada funcionário é identificado por meio de um número de matrícula, sendo que esse número de matrícula representa um valor acurado que já foi analisado na operação de limpeza.

Existem diversos conflitos de valores de atributos nos dados sendo integrados. Exemplos incluem os valores para os atributos de sexo e nome do funcionário. Considerando o sexo, tem-se as seguintes possibilidades: F/M; Feminino/Masculino; e 0/1. Considerando o nome, tem-se as seguintes possibilidades: Adenildo Campos; Campos, Adenildo; e A. Campos.

3.6 CARGA

Após as operações realizadas, os dados são armazenados no *data mart*, de acordo com o esquema de relação *funcionario*. Os dados tornam-se disponíveis, portanto, para consulta e análise dos usuários de suporte à decisão.

4 CONCLUSÃO

Neste texto, foram descritos os seguintes conceitos e aspectos relacionados:

- Contextualização: características do projeto de *data warehousing*, impacto da manipulação de *big data* no processo ELT e diferença entre os conceitos de instância e esquema.
- Detalhamento das operações do processo de ETL/ELT: extração, tradução, limpeza, integração e carga.
- Exemplificação das operações do processo de ETL usando como base a aplicação de *data warehousing* da **BI Solutions**.

Referências

[1] J. Bleiholder and F. Naumann. Conflict handling strategies in an integrated information system. In *Proceedings of the International Workshop on Information Integration on the Web*, 2006.

[2] J. Bleiholder and F. Naumann. Data fusion. 41(2):1:1–1:41, 2009.

[3] R. A. Capozzi. Lei Geral de Proteção de Dados no Brasil - Uma abordagem prática. In *Slides apresentados no 10 Congresso RTI de Provedores de Internet 2019 e 12 Congresso RTI de Data Centers*, 2019.

[4] M. Chen, S. Mao, , and Y. Liu. Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209, 2014.

[5] X. L. Dong and D. Srivastava. Big data integration. *Proceedings of the VLDB Endowment*, 6(11):1188–1189, 2013.

[6] R. Elmasri and S. B. Navathe. *Sistemas de Bancos de Dados*. Pearson, 4th edition, 2011.

[7] A. Gupta and I.S. Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Engineering Bulletin*, 18(2):3–18, 1995.

[8] A. Kadadi, R. Agrawal, C. Nyamful, and R. Atiq. Challenges of data integration and interoperability in big data. In *Proceedings of the 2014 IEEE International Conference on Big Data*, pages 38–40, 2014.

[9] A. C. R. B. Portes. Modelagem conceitual de processo de ETL: Uma abordagem baseada em operadores. Master's thesis, Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de São Carlos, 2020.

[10] S. Sharma and V. Mangat. Technology and trends to handle big data: Survey. In *Proceedings of the Fifth International Conference on Advanced Computing Communication Technologies*, pages 266–271, 2015.

[11] Alkis Simitsis, Panos Vassiliadis, and Timos Sellis. Extraction-transformation-loading processes. pages 240–245, 01 2005.

[12] S. Spaccapietra and C. Parent. View integration: A step forward in solving structural conflicts. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):258–274, 1994.

[13] R. Wrembel. Novel big data integration techniques: Painel discussion at BigNovelTI 2017@ADBIS2017. 2017.