Capítulo

2

Procedência dos dados

Neste trabalho, entende-se por procedência dos dados o conjunto de metadados que possibilita identificar a fonte dos dados bem como os processos de transformação aplicados aos dados, desde a sua criação até seu estado atual. Na área de computação, questões sobre a procedência dos dados têm se tornado cada vez mais freqüentes, devido principalmente à grande quantidade de dados disponíveis e à necessidade de avaliar a qualidade e/ou veracidade dos mesmos.

O objetivo deste capítulo é descrever o estado da arte no tema procedência dos dados, discutindo os principais aspectos relacionados a esta área de pesquisa. O capítulo aborda diversos trabalhos sobre procedência que têm sido propostos na literatura, considerando cenários distintos e funcionalidades específicas.

Este capítulo está organizado da seguinte maneira. Na Seção 2.1 são discutidas algumas motivações para o armazenamento dos dados de procedência. Em seguida, na Seção 2.2 são considerados alguns aspectos sobre como realizar a manutenção dos dados de procedência. Por fim, na Seção 2.3 são feitas algumas considerações finais sobre o assunto abordado neste capítulo.

2.1 Motivações para Armazenar a Procedência dos Dados

Existem diversas motivações para se armazenar a procedência dos dados (Simmhan et al., 2005b; Tan, 2004), dentre as quais destacam-se:

- Garantia da qualidade dos dados;

- Verificação dos dados;
- Auditoria e autoria dos dados;
- Manutenção de bancos de dados;
- Informação sobre os dados;
- Reenvio de dados para a fonte;
- Análise de tendências; e
- Reprodução de experimentos científicos.

As Seções 2.1.1 a 2.1.8 detalham respectivamente cada motivação apresentada.

2.1.1 Qualidade dos dados

Muitos bancos de dados são formados por meio da coleta de dados de diversas fontes, juntamente com a inserção manual de dados por parte de seus usuários. Um bom projeto de um banco de dados visa armazenar esses dados de maneira consistente. Entretanto, quando um usuário requisita o acesso aos dados desse banco, ele pode estar interessado não somente em um determinado dado, mas também na origem desse dado para que possa estimar a sua confiabilidade. Contudo, em projetos convencionais, não é possível assegurar a qualidade dos dados armazenados, pois nenhuma informação a esse respeito é armazenada.

A informação da procedência fornece um meio de estimar a qualidade dos dados. Se a fonte é confiável, pode-se inferir que o dado obtido dessa fonte tem uma maior probabilidade de estar correto do que um dado de fonte desconhecida, ou até mesmo de um dado cuja fonte é conhecida, mas não-confiável. Dessa maneira, o usuário pode filtrar dados de maior qualidade conforme a confiabilidade das fontes.

2.1.2 Verificação dos dados

Devido à grande disponibilidade de dados na *web*, tem se tornado cada vez mais comum referenciar dados que se encontram na rede. Contudo, diferentemente de livros, páginas de internet sofrem alterações ao longo do tempo. Ao ocorrer uma mudança em uma página, seu conteúdo pode deixar de validar alguma citação feita anteriormente. Desta maneira, o documento que referencia a página se torna inconsistente com sua referência.

A fim de manter essa consistência, é necessário que cada versão das informações referenciadas seja armazenada, de tal forma que as referências continuem consistentes ao longo do tempo, independentemente se a página referenciada sofrer alterações. Nesse sentido, procedência dos dados vai além do conceito de apenas armazenar informações

sobre fontes de dados e processos que transformam dados, e passa a englobar outros conceitos como versionamento de dados ao longo dos tempos, com o intuito de que os dados possam ser rastreados desde a sua criação.

2.1.3 Auditoria e Autoria dos dados

Tanto no meio empresarial quanto acadêmico, processos de auditoria dos dados consistem em verificar, por exemplo, se os dados estão sendo produzidos corretamente e se relatórios estão sendo gerados a partir das fontes corretas, dentre outras coisas. Nesse contexto, a informação sobre a procedência dos dados pode ser utilizada para a realização de processos de auditoria dos dados. Quando dados são constatados como incorretos, por meio da procedência é possível recuperar o fluxo dos dados até chegar ao banco de dados. Analisando os processos de transformação aplicados a esses dados durante esse fluxo e suas respectivas fontes, pode-se determinar qual etapa resultou em dados incorretos.

Ainda na área de auditoria, a procedência dos dados pode ser útil na determinação do autor de um dado e então atribuir-lhe a responsabilidade pela geração do mesmo. Por exemplo, em artigos científicos todo tipo de asserção deve ser validada por algum experimento. Quando o autor não é o proprietário e/ou responsável pela asserção sendo feita, ele deve referenciar a fonte de onde extraiu tal informação. Em ambientes nos quais os dados fluem de um sistema para outro, e os dados são compartilhados entre diversos pesquisadores, manter a procedência dos dados pode ajudar na identificação do autor de um determinado dado e na atribuição de crédito pela autoria, referenciando-o.

2.1.4 Manutenção de bancos de dados

Em geral, fontes de dados sofrem diversas alterações ao longo do tempo. Novos dados são inseridos, outros removidos e outros modificados. Um banco de dados que integra dados de diversas fontes pode se tornar obsoleto conforme os dados das fontes mudam.

Mantendo a informação da procedência dos dados, é possível voltar às fontes e verificar eventuais alterações e atualizar o banco de dados integrado com as novas versões dos dados. Ainda como forma de manutenção de bancos de dados, a informação sobre a procedência pode ser utilizada para duplicar um banco de dados, ou mesmo recriá-lo, se necessário.

2.1.5 Informação sobre os dados

A procedência dos dados pode ser utilizada por motivo informacional para contextualizar um dado. Por exemplo, ter conhecimento dos motivos que geraram a necessidade de um dado estar armazenado, saber quando ele foi recuperado e quem o recuperou, são informações que ajudam na contextualização de um dado.

Um cenário típico para essa motivação consiste em pesquisa bibliográfica, na qual é comum a recuperação de diversos artigos de uma só vez. Realizar anotações sobre a importância de cada artigo no contexto do trabalho ajuda o pesquisador no entendimento desse artigo, principalmente quando o mesmo é consultado depois de um longo período após a sua aquisição. Ademais, é possível saber quais fontes já foram pesquisadas anteriormente e quais artigos já foram recuperados de cada fonte.

2.1.6 Reenvio de dados para a fonte

No cenário de integração de dados, diversas fontes são consultadas para extração de dados. O processo de integração pode retificar alguns dados e/ou identificar outros como incorretos ou omissos em algumas fontes. Quando dados são identificados como incorretos ou omissos, eles devem ser prontamente corrigidos. Em um processo de auditoria, pode ser possível alterar diretamente a fonte dos dados se a procedência dos mesmos estiver armazenada.

2.1.7 Análise de tendências

Em geral, com o passar do tempo a quantidade de dados em um banco de dados tende a aumentar. Se a informação de procedência é armazenada, a quantidade dessas informações também tende a crescer. É possível então utilizar essas informações sobre procedência para análise de tendência e avaliar, por exemplo, qual é o custo-benefício na extração dos dados de cada fonte.

Um exemplo prático pode ser encontrado nas universidades que obtêm dados do banco de dados do currículo Lattes. Para obter os currículos de seus pesquisadores por meio do *Lattes Extrator*, uma Instituição (ex: uma Universidade) deve pagar uma taxa anual (Lattes, 2007). Valendo-se da informação de procedência dos dados, a Instituição pode averigüar se os dados recuperados do *Lattes Extrator* estão sendo efetivamente utilizados e, caso isso não esteja acontecendo, a Instituição pode deixar de utilizar os serviços e economizar recursos financeiros.

2.1.8 Reprodução de experimentos científicos

Artigos científicos consistem no registro do conhecimento que está sendo exposto, juntamente com resultados de experimentos realizados. Em geral, em um artigo são expostos métodos, algoritmos e arquiteturas, os quais são validados por experimentos e comparação com trabalhos correlatos. Para um cientista comparar o trabalho desenvolvido com trabalhos correlatos, ele precisa executar os mesmos experimentos com todos os trabalhos que deseja comparar. Esse cenário ocorre em diversas áreas do conhecimento, não estando restrito apenas à área de computação.

Porém, muitas vezes o cientista não tem à sua disposição as informações que necessita para reproduzir os experimentos realizados por outros cientistas, mas apenas as informações relatadas nos artigos. Ademais, muitas vezes o próprio cientista que realizou o experimento não é capaz de reconstituir todo o processo efetuado, parâmetros utilizados, dados utilizados como entrada, dentre outras coisas necessárias para a reprodução exata do experimento. Nesse sentido, a procedência dos dados para esse cenário tem como objetivo armazenar o ambiente em que o experimento foi realizado, englobando os dados utilizados como entrada e os dados gerados como saída, os processos/aplicativos utilizados na geração dos resultados e os parâmetros, para que o experimento seja reproduzível. Dessa maneira, em um ambiente colaborativo, qualquer cientista pode fazer uso das informações de procedência para reproduzir um determinado experimento.

2.2 Aspectos Relacionados à Procedência dos Dados

Diversos aspectos devem ser considerados para o desenvolvimento de técnicas de procedência dos dados. A Figura 2.1 ilustra esses aspectos e destaca a seqüência na qual ocorrem.

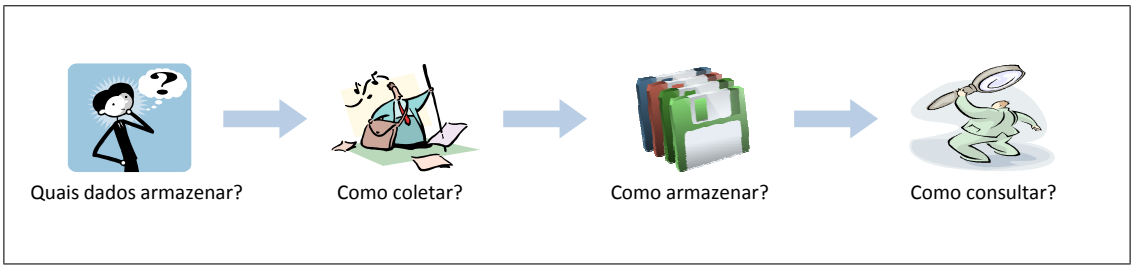

Figura 2.1: Aspectos da procedência dos dados.

Assim como ilustrado na Figura 2.1, o primeiro aspecto a ser considerado trata-se da definição de quais dados devem ser armazenados bem como a granularidade desses dados (Seção 2.2.1). Após a definição dos dados a serem armazenados, é preciso estabelecer uma estratégia de coleta dos dados de procedência (Seção 2.2.2). Na seqüência, é necessário armazenar os dados para torná-los acessíveis futuramente (Seção 2.2.3). Por fim, é necessário tornar os dados de procedência disponíveis para que os usuários possam consultar os dados armazenados (Seção 2.2.4).

2.2.1 Quais dados de procedência armazenar?

Primeiramente é necessário definir quais dados devem ter sua procedência monitorada. Isso deve levar em consideração o custo despendido para manutenção desses dados e o benefício alcançado posteriormente. Após definidos os dados que serão monitorados, o

próximo passo a ser considerado em um projeto de armazenamento da procedência dos dados é decidir:

- que tipos de dados de procedência devem ser armazenados; e
- qual a granularidade dos dados sobre a procedência.

Tipos de dados de procedência

Devido ao fato da área de pesquisa sobre procedência dos dados ser relativamente nova, ainda não há um consenso sobre a taxonomia a ser utilizada para classificar os tipos de dados que podem ser armazenados. As taxonomias existentes na literatura são: *source* e *transformation provenance*, *provenance meta-information* e *process meta-information*, *prospective* e *retrospective provenance*, *why-* e *where-provenance*, e *when-how-what components*.

Segundo Glavic e Ditt (2007), a procedência dos dados pode ser armazenada com dois enfoques principais: (1) na fonte dos dados e (2) nos processos de transformação pelos quais os dados foram submetidos. O primeiro enfoque é definido como *source provenance* e o segundo como *transformation provenance*. *Source provenance* ainda é subdividido em três partes: *input source*, *original source* e *contributing source*. Trata-se de uma divisão que separa, respectivamente, (1) as fontes de dados que foram consultadas para a geração de um dado, (2) os itens de dados que realmente contribuíram para a geração de um dado, e (3) o conjunto mínimo de itens de dados necessários para a geração de um dado. *Source* e *transformation provenance* são complementares, sendo que esses dois enfoques podem co-existir em um mesmo sistema de procedência dos dados.

Para ilustrar os conceitos de *source* e *transformation provenance*, considere uma aplicação que armazena dados oriundos de diferentes bancos de dados (i.e., fontes) em um único banco de dados integrado. Desde que estas fontes podem conter dados redundantes, possíveis heterogeneidades decorrentes dessa redundância têm que ser resolvidas no processo de integração. Quando um usuário consulta o banco de dados integrado, os dados obtidos como resposta são aqueles que foram considerados como corretos no processo de integração. Sendo assim, o usuário também pode estar interessado na qualidade do resultado obtido. Se um determinado dado contido na resposta foi coletado e inserido sem nenhum tipo de transformação, a informação sobre a sua fonte pode fornecer um meio de assegurar a sua qualidade. Esse tipo de procedência é denominado *source provenance*. Caso contrário, é possível validar a transformação realizada por meio de informações tais como quais transformações automáticas ou manuais foram efetuadas sobre os dados originais. O processo de integração é um exemplo de transformação que pode ser efetuada sobre os dados. Esse tipo de procedência é denominado *transformation provenance*.

Del Rio e Silva (2007) definem os conceitos *provenance meta-information* e *process meta-information*. *Provenance meta-information* diz respeito à informação sobre a fonte

de dados que originou o dado, sendo diretamente relacionado ao conceito de *source provenance*. Já o conceito *process meta-information* refere-se aos processos que originaram um dado, e constitui um conceito semelhante ao de *transformation provenance*. Porém, *process meta-information* é mais abrangente no sentido que captura todo o ambiente envolvido na criação do dado, e não apenas as transformações aplicadas aos dados originais.

Já Zhao et al (2006) definem os conceitos *prospective* e *retrospective provenance*. Esses conceitos estão relacionados aos conceitos de *source* e *transformation provenance*, respectivamente. Contudo, as definições feitas por Zhao et al (2006) são mais abrangentes. *Prospective provenance* diz respeito a todos os aspectos relacionados ao processo para criação de um dado, enquanto que *retrospective provenance* diz respeito ao ambiente de execução do processo, bem como as fontes de dados utilizadas.

Buneman et al (2001) definem os conceitos *why-* e *where-provenance*. A motivação para tal classificação é a distinção entre a fonte de origem de um determinado dado (i.e., *where-provenance*) e porque o dado se encontra no banco de dados (i.e., *why-provenance*). Enquanto *why-provenance* armazena todas as fontes de dados que contribuíram para a geração de um dado, *where-provenance* armazena os dados das fontes que realmente contribuíram para a geração de um dado. *Why-* e *where-provenance* definem conceitos relacionados aos conceitos *original* e *contributing source*, respectivamente.

A taxonomia *when-how-what components* é proposta em Widow (2005). O componente ***when*** indica o momento de criação do dado. O componente ***how*** considera diversas maneiras de inserção de dados no banco, tais como por meio de consultas, por aplicações externas que acessam o banco, por atualizações efetuadas nos dados, por meio de importação de dados de outra fonte ou por carregamento em lote. O terceiro componente, ***what***, indica quais dados das fontes foram utilizados para derivar o dado atual.

Granularidade dos Dados

Os dados sobre a procedência podem ser armazenados em diversos níveis de detalhe, ou seja, em diversas granularidades. Granularidade dos dados e nível de detalhe são conceitos inversamente proporcionais, ou seja, quanto maior a granularidade, menor o nível de detalhe e quanto menor a granularidade, maior o nível de detalhe. A título de exemplificação, considere um banco de dados relacional. Os dados sobre procedência podem ser armazenados em nível de tabela (maior granularidade), de tupla (média granularidade) ou então de atributo (menor granularidade) (Glavic e Ditt, 2007; Widom, 2005). O conceito de granularidade aplicado à procedência é o mesmo encontrado na teoria de *data warehouse* (Chaudhuri e Dayal, 1997; Kimball e Ross, 2002; Wu e Buchmann, 1997).

Três aspectos diferentes estão relacionados à granularidade dos dados. O primeiro aspecto refere-se ao espaço de armazenamento dos dados, o qual é inversamente proporcional à granularidade dos dados coletados. Sendo assim, quanto menor a granularidade, maior o espaço necessário para armazenar os dados. O segundo aspecto, custo da coleta dos dados, também é inversamente proporcional à granularidade (Simmhan et al., 2005a). Já o terceiro aspecto refere-se à variedade de consultas que podem ser respondidas. Quanto maior o nível de detalhe, mais consultas podem ser respondidas sobre a procedência dos dados, em especial perguntas de caráter mais específico.

Por exemplo, em um SGBD, se a granularidade estiver em nível de tupla, podem ser respondidas consultas em nível do SGBD como um todo, em nível de tabela ou então em seu nível mais detalhado, as tuplas. Porém, a quantidade de dados de procedência em nível de tupla requer maior espaço de armazenamento e maior tempo de coleta do que dados de procedência em nível de tabela ou do SGBD como um todo. Dessa forma, se a granularidade for armazenada em nível de tupla, maior será o tempo de coleta e o espaço de armazenamento necessário, entretanto, uma maior variedade de consultas poderá ser respondida.

Outro ponto a ser destacado é que os dados sobre a procedência, em uma visão simplista, podem ser considerados um misto de *log* e versionamento, ou o versionamento de um *log*. Por se tratar de um versionamento, dados antigos são mantidos para que se possa rastrear a história de um dado. Contudo, existem alguns pontos a serem considerados para se realizar o versionamento. Se a informação de procedência desejada for muito rica em detalhes (baixa granularidade), o crescimento do banco de dados de procedência pode se tornar um fator proibitivo para o armazenamento de dados históricos. Surge, então, a necessidade de se elaborar políticas de manutenção da procedência sobre dados obsoletos (Goble, 2002; Muniswamy-Reddy et al., 2006). Por exemplo, depois que um dado for removido, o sistema de procedência deve manter sua informação armazenada? Por quanto tempo? Quais dados sobre a procedência devem ser mantidos? Todos? Essas são algumas questões em aberto que precisam de estudos mais aprofundados.

Portanto, a granularidade é um ponto que deve ser definido cuidadosamente em um projeto sobre procedência, a fim de evitar a necessidade de uma maior quantidade de espaço de armazenamento e manter o tempo de coleta no menor custo possível. Assim, é preciso escolher os dados de maior relevância para manter a procedência.

2.2.2 Como coletar os dados de procedência?

Depois de identificados os dados a serem armazenados, é necessário elaborar uma estratégia para a coleta dos mesmos. A coleta dos dados de interesse pode ser realizada manual ou automaticamente, usando as abordagens *lazy* ou *eager*.

Coleta manual e automática

A coleta dos dados de procedência pode ser feita com ou sem a intervenção do usuário. A estratégia a ser adotada varia caso a caso. Por exemplo, em bancos de dados curados manualmente, o usuário pode buscar dados de diversas fontes, utilizando mais do que uma ferramenta de busca. Deste modo, a coleta da procedência sobre esses dados pode ser feita diretamente pelo usuário. Porém, a tarefa de coleta da procedência é secundária frente ao interesse do usuário em realizar pesquisas sobre assuntos específicos. Sendo assim, atribuir ao usuário a tarefa de coleta dos dados acarreta a introdução de potenciais pontos de erros (Buneman et al., 2006a). Por isso, estratégias automáticas são preferidas às manuais.

Nos casos em que a aplicação oferece suporte à coleta de dados, a procedência pode ser implementada de maneira transparente para o usuário. Os trabalhos de Munroe et al (2006), Del Rio e Silva (2007) e Muniswamy-Reddy et al (2006) abordam o problema de coleta de dados de maneira automática. A principal diferença entre esses trabalhos refere-se ao fato da coleta ser realizada pelas aplicações utilizadas pelos usuários, por serviços externos às aplicações, ou pelo próprio sistema operacional.

Munroe et al (2006) propõem o *Provenance Incorporating Methodology* (PrIMe), uma metodologia de engenharia de *software* para o desenvolvimento de aplicações que ofereçam suporte à coleta da procedência dos dados. Essa metodologia divide o processo de desenvolvimento em três fases. A primeira fase consiste em identificar os casos de uso da procedência e os itens de dados associados a cada caso. Na segunda fase identificam-se os atores envolvidos nos casos de uso e suas interações com a aplicação, além de quais itens de dados são considerados em cada interação entre os atores. Por ator entende-se toda entidade que interage com a aplicação, tais como um *web service* ou uma pessoa. O terceiro e último passo é fazer as devidas alterações na aplicação de modo que ela ofereça suporte à coleta da procedência. A vantagem desta metodologia é que a aplicação é capaz de coletar a procedência internamente e de maneira transparente para o usuário. Em contrapartida, se os passos propostos não forem aplicados durante o desenvolvimento da aplicação, a alteração da aplicação para oferecer suporte à procedência pode se tornar custosa a ponto de inviabilizar o processo.

Del Rio e Silva (2007) consideram a coleta da procedência por meio de serviços externos à aplicação. Mais especificamente, os autores fazem uso da *PLM Service Wrapper* (PSW). Esse serviço coleta informações de entrada e saída das aplicações que participam de um *workflow*, interceptando mensagens e eventos gerados pelas aplicações. Essa abordagem não introduz complexidade extra à aplicação. Porém, os dados que são internos à aplicação e não fazem parte dos dados de entrada/saída, não podem ter a procedência coletada.

Muniswamy-Reddy et al (2006) propõem um sistema para a coleta da procedência em nível de sistema operacional e seu sistema de arquivos. A procedência é coletada

em nível de arquivos, traduzindo seqüencias de comandos do sistema operacional em informações sobre procedência. A informação da procedência coletada consiste em uma descrição dos processos executados para gerar um arquivo. Assim como a coleta por meio de serviços externos à aplicação, implementar a procedência em nível de sistema operacional tem a vantagem de tornar a coleta transparente tanto para o usuário final quanto para o desenvolvedor de aplicações. Porém, uma desvantagem dessa abordagem é que a procedência é coletada com maior granularidade (em nível de arquivos do sistema de arquivos), pois a coleta de informações mais específicas depende do fato das aplicações oferecerem suporte à procedência.

Abordagens Lazy x Eager

A abordagem *lazy* (i.e., preguiçosa) consiste em coletar a procedência de um dado apenas quando esta for requisitada. Essa abordagem considera que existe uma consulta e uma fonte de dados, além de assumir que ambas (a consulta e a fonte) estão disponíveis no momento da coleta da procedência. Em geral, essa abordagem é utilizada em SGBDs da seguinte maneira: dado uma consulta Q e sua saída d, gera-se outra consulta cuja finalidade é obter os itens de dados que em conjunto com a consulta Q resultaram no dado d (Tan, 2004). Nessa abordagem, o consumo de memória é pouco, ou quase nenhum, uma vez que as informações são calculadas sempre que necessário. Porém, a informação sobre procedência recuperada é restrita somente ao contexto do SGBD, não sendo possível recuperar informações externas ao mesmo.

Já a abordagem *eager* (i.e., gulosa), consiste em armazenar a procedência dos dados conforme o dado passa de um sistema para outro ou sofre transformações. Essa abordagem também é referenciada na literatura por *annotation* (i.e., anotação). No momento em que a procedência é requisitada, pode ser que a fonte de dados não esteja mais acessível e/ou a consulta não tenha sido documentada e ainda assim, na abordagem *eager*, é possível recuperar a procedência dos dados. Essa vantagem é decorrente do fato de que, nessa abordagem, a procedência é armazenada conforme os dados são gerados. Ademais, essa abordagem permite um nível de detalhe maior que a abordagem *lazy*. Contudo, as desvantagens da abordagem *eager* consistem no aumento da complexidade das aplicações em manter anotações sobre os dados e no espaço adicional necessário para o armazenamento de tais anotações.

Para ilustrar as abordagens *lazy* e *eager*, considere um sistema de procedência de dados que coleta dados em nível de sistema operacional. Esse sistema coleta a procedência conforme os processos são executados e dados (por exemplo, arquivos) são gerados. Dependendo do nível de detalhe coletado, o ambiente (*software* e *hardware*) no qual o processo foi executado pode alterar com o passar do tempo. Nesse caso, com o emprego da abordagem *lazy*, quando a informação sobre a procedência for requisitada, ela não mais refletirá o ambiente em que o processo realmente foi executado. Já com o emprego da

abordagem *eager* a informação sobre a procedência será corretamente recuperada, uma vez que ela é coletada no momento em que o processo é executado e os dados são gerados (Muniswamy-Reddy et al., 2006).

Os trabalhos que abordam o problema de coleta de dados de maneira automática (i.e., Munroe et al (2006), Del Rio e Silva (2007) e Muniswamy-Reddy et al (2006)) enquadram-se como abordagens *eager*. Nesses trabalhos, os dados de procedência são coletados conforme são gerados.

2.2.3 Como armazenar os dados coletados?

Assim como toda a área de procedência, aspectos relacionados ao armazenamento dos dados ainda se encontram em desenvolvimento. Devido ao fato da área de procedência abranger praticamente todas as formas de manipulação e armazenamento de dados, os trabalhos existentes na literatura enfocam a procedência em nível de sistema operacional (Muniswamy-Reddy et al., 2006; Simmhan et al., 2005a), em nível de aplicativo (Bose e Frew, 2004), em nível de serviço externo ao aplicativo (Del Rio e Silva, 2007) e em nível de SGBD (Buneman et al., 2006a,b; Widom, 2005). Ademais, Buneman (2006) et al fazem algumas considerações adicionais sobre como armazenar os dados de procedência.

Considerando o cenário no qual os dados são coletados em nível de sistema operacional, mais especificamente no sistema de arquivos, Simmhan (2005a) et al argumentam que a informação sobre a procedência pode ser armazenada dentro do arquivo ou em um banco de dados de procedência, por exemplo. Uma forma de armazenar a procedência dentro do arquivo é incluir metadados no cabeçalho do mesmo. Desta forma, a procedência fica fortemente acoplada ao dado e o acompanha quando este é copiado/movido para outros locais. Porém, nessa abordagem, fica difícil procurar por dados que satisfaçam a algum critério de procedência, uma vez que a informação encontra-se espalhada por diversos arquivos.

O trabalho de Muniswamy-Reddy et al (2006) propõe um sistema de procedência que coleta os dados diretamente no sistema de arquivos. Para o armazenamento dos dados é utilizado um SGBD em nível do *kernel* do sistema operacional. Essa abordagem tem como vantagem a rapidez na busca de arquivos com base em sua procedência, uma vez que os dados de procedência estão concentrados em um SGBD. Contudo, a manutenção dos dados de procedência quando arquivos são criados, alterados ou removidos é mais complexa do que se a procedência estivesse dentro do arquivo.

Em nível de aplicativo, a procedência dos dados pode ser gerenciada em formatos de arquivos específicos. Por exemplo, para o formato XML, pode-se adaptar seu XML *schema* para o armazenamento da procedência dos dados, ou pode-se criar um XML *schema* próprio para os dados da procedência. O trabalho de Bose (2004) et al (Bose e Frew,

2004) é um exemplo de trabalho que propõe um XML *schema* próprio para armazenar a procedência de dados advindos de pesquisas ambientais.

Del Rio e Silva (2007) fazem uso da *Proof Markup Language* (PML) para armazenar a informação sobre a procedência. A PML é uma linguagem com base na *Web Ontology Language* (OWL), projetada para facilitar interoperabilidade e confiabilidade na *web* semântica (da Silva et al., 2006). Essa linguagem deve ser utilizada em conjunto com o PSW, apresentado na Seção 2.2.2, pois o PSW formata os dados coletados na linguagem PML. Porém, testes realizados no trabalho de Del Rio e Silva (2007) mostraram que a interpretação de arquivos nessa linguagem é muito cara computacionalmente, prejudicando assim a escalabilidade do sistema.

SGBDs constituem outro cenário bastante comum para uso de técnicas de procedência dos dados. Com a crescente disponibilidade de informações na internet, bancos de dados curados manualmente têm sido mantidos a fim de reunir dados precisos sobre determinados assuntos, assim como os relacionados à bioinformática (Buneman et al., 2006a,b). Nesse cenário, a informação da procedência pode ser armazenada juntamente com o banco de dados em questão, sendo implementado, por exemplo, como um esquema do banco de dados. Desta maneira, a informação da procedência pode ser facilmente vinculada aos dados. Contudo, essa abordagem aumenta a complexidade do banco de dados, uma vez que introduz mais relações e relacionamentos.

No SGBD Trio (Widom, 2005), cada banco de dados possui uma relação dedicada ao armazenamento da procedência, denominada *lineage*. Essa relação armazena uma referência para a tupla alvo (que está em alguma das relações do banco de dados), um identificador para a maneira como a tupla foi derivada, um identificador de como a tupla foi derivada, e uma referência aos dados, ou os próprios dados, que derivaram a tupla. A granularidade dos dados no sistema Trio pode ser feita em nível de tupla ou relação, conforme os requisitos da aplicação.

Uma abordagem alternativa à criação de uma relação dedicada é criar um banco de dados separado para os dados de procedência. Essa estratégia é utilizada em (Zhao et al., 2006) e (Buneman et al., 2006b). Assim, o projeto do banco de dados fica modularizado, sendo um banco para os dados e outro para a procedência. Ademais, a complexidade do banco de dados fica restrita à aplicação. Como nem sempre um SGBD é a melhor escolha para todos os casos, uma vantagem dessa abordagem está no fato de que o SGBD para procedência pode ser escolhido conforme a necessidade. Contudo, a manutenção da procedência dos dados se torna mais complexa, uma vez que não é possível fazer uma ligação direta entre o dado e sua procedência.

Independentemente do nível no qual a procedência é tratada (i.e., em nível de sistema operacional, em nível de aplicativo, em nível de serviço externo ao aplicativo ou em nível de SGBD), um aspecto importante a ser considerado refere-se a como armazenar os dados de forma a diminuir o espaço de armazenamento necessário. Em (Buneman

et al., 2006b) são propostas quatro técnicas para armazenamento. A primeira técnica, denominada *naive provenance*, não considera nenhum tipo de otimização, armazenando o máximo de dados possível, conseguindo desta maneira o maior nível de detalhe. A segunda técnica, denominada *transactional provenance*, propõe agrupar os dados de procedência em transações. Por meio das transações podem ser eliminados dados desnecessários. Por exemplo, considere que um usuário copia um dado de uma fonte A, em seguida sobrescreve esse dado com outro de uma fonte B, e por fim encerra a transação. Os dados de procedência que são armazenados nesse caso são aqueles advindos da fonte B. A terceira técnica, denominada *hierarchical provenance*, introduz o conceito de pai/filho para dados e propõe que os dados da procedência de um dado filho podem ser recuperados pela procedência de seu pai. Dessa maneira, não é preciso duplicar para o dado filho a procedência que está no dado pai. A vantagem dessa técnica é que todos os dados são armazenados, ao contrário da segunda técnica. A quarta técnica, denominada *transactional-hierarchical provenance*, é uma combinação da segunda e da terceira técnicas. Essa quarta técnica armazena todos os dados de procedência e os representa da maneira mais concisa dentre as quatro técnicas propostas.

2.2.4 Como consultar os dados armazenados?

Com os dados sobre a procedência armazenados, pode-se investigar estratégias para consultar esses dados. Esta seção primeiro descreve os tipos de consulta existentes na literatura e em seguida relaciona trabalhos existentes com relação a esses tipos de consulta.

Tipos de consulta

Existem dois tipos de consulta que podem ser realizados sobre os dados de procedência:

- tipo um: pesquisar os dados, e a partir do resultado obtido verificar a sua procedência; ou
- tipo dois: recuperar os dados que satisfazem a um determinado critério de procedência (e.g., dados que foram criados/inseridos no banco no mês de outubro).

Cada um dos dois tipos de consulta enfoca diferentes aspectos na busca dos dados. Enquanto o primeiro visa buscar os dados e depois, se necessário, garantir a qualidade dos mesmos, o segundo oferece um método mais direto para buscar dados com base em sua qualidade/procedência. Assim sendo, os dois tipos de consultas consistem em abordagens complementares e podem co-existir em um mesmo sistema de procedência.

A viabilidade ou não de um sistema permitir os dois tipos de consulta depende principalmente de como os dados estão armazenados. Os sistemas de procedência que armazenam os dados em nível de arquivo do sistema operacional, mais especificamente

dentro do arquivo como metadado, dificultam consultas do tipo dois. Em consultas desse tipo para esse cenário, todos os arquivos do sistema de arquivos devem ser consultados, o que demanda grande quantidade de recursos e tempo, prejudicando a escalabilidade do sistema (Muniswamy-Reddy et al., 2006).

Estratégias para consulta e visualização dos dados

Alguns trabalhos existentes na literatura que propõem métodos para consulta dos dados de procedência envolvem o acesso aos dados por meio de uma interface ou API (*Application Programming Interface*), de modo que as aplicações possam acessar a procedência e tratar os resultados da maneira mais adequada (Muniswamy-Reddy et al., 2006). Outros desenvolvem ferramentas para consultas em conjunto com linguagens próprias para acesso aos dados, por exemplo (Widom, 2005). Com relação à visualização, esta pode ser feita por meio de grafos que descrevem a história de um dado. Del Rio e Silva (2007) propõem um trabalho que utiliza grafos como forma de consulta e visualização dos dados de procedência. Todos esses trabalhos oferecem suporte aos tipos de consulta um e dois.

Muniswamy-Reddy et al (2006) propõem um método de consulta da procedência por meio de uma ferramenta, e uma interface de acesso similar à de um banco de dados. A ferramenta proposta permite a navegação pelos arquivos do sistema operacional, tal qual um navegador de arquivos, e permite que o usuário recupere a procedência de um dado arquivo. O usuário também pode procurar por arquivos utilizando uma consulta aos dados de procedência. Nesse sistema os dados de procedência estão centralizados em um sistema próprio de armazenamento, ao invés de estarem no próprio arquivo.

No sistema Trio, Widom (2005) propõe uma extensão da linguagem SQL (*Structured Query Language*), denominada TriQL, para consultar os dados sobre procedência. Dois principais fatores motivaram o uso da linguagem SQL como base da linguagem de consulta do sistema Trio. Primeiro, o fato de a linguagem SQL ser amplamente difundida entre usuários de SGBDs, o que facilita o aprendizado e a disseminação da linguagem proposta. Segundo, a linguagem SQL oferece suporte a diversas funcionalidades requisitadas pelo sistema Trio, tais como funções de agregação e agrupamento dos dados e operadores como o *like*, dentre outros. A abordagem de armazenamento da procedência proposta no sistema Trio (Seção 2.2.3) permite que os tipos de consulta um e dois sejam realizados facilmente. A consulta do tipo um, que para SGBDs trata-se da obtenção dos dados de procedência de uma tupla de uma relação R do banco, pode ser feita com uma operação de junção da relação R com a relação que armazena a procedência. Já a consulta do tipo dois, que para SGBDs trata-se de selecionar dados que satisfazem a um critério de procedência, pode ser feita com uma operação de seleção na tabela de procedência, seguida de uma operação de junção com a relação que armazena os dados.

Del Rio e Silva (2007) propõem o sistema *Probe-It!* para visualização dos dados de procedência em *workflows*. O objetivo é unir as técnicas de procedência e visualização dos

dados e oferecer um sistema que facilite a análise e o entendimento de grandes quantidades de dados de procedência. Esse sistema separa as tarefas de coleta e armazenamento dos dados de procedência da tarefa de consulta e visualização da procedência. Para uso do sistema, assume-se que existe um banco de dados de procedência acessível como fonte, a qual deve ser passada como parâmetro pelo usuário do sistema. A principal característica desse trabalho é gerar visualizações por meio de grafos para as justificativas dos dados obtidos como resultados de consultas. Ademais, essas visualizações podem ser comparadas entre diferentes resultados de consultas.

2.3 Considerações Finais

Neste capítulo foi descrito o estado da arte no tema procedência dos dados. Primeiramente, foram investigadas diversas motivações para se armazenar a procedência. Na seqüência, foi realizada uma classificação de diversos aspectos que devem ser levados em consideração para a proposta de um modelo de procedência dos dados. Estes aspectos foram classificados em: (i) quais dados de procedência armazenar, (ii) como coletar os dados de procedência, (iii) como armazenar os dados coletados; e (iv) como consultar os dados armazenados.

Para cada um dos aspectos identificados neste capítulo, foram referenciados diversos trabalhos que têm sido propostos na literatura. Entretanto, estes trabalhos foram apresentados resumidamente, desde que o objetivo do capítulo foi apresentar uma visão geral sobre a área de pesquisa em procedência dos dados. Em contrapartida, o Capítulo 3 descreve detalhadamente dois diferentes sistemas que provêm modelos de procedência dos dados, e que são mais relacionados ao objetivo do trabalho a ser desenvolvido.