2.1. Integração dos Dados

Integração dos dados é um problema enfrentado por aplicações que precisam acessar várias fontes de dados autônomas e heterogêneas [Hsiao and Kamel 1989, Sheth and Larson 1990, Hsiao 1992, Aguiar 1995, Halevy et al. 2005, Halevy et al. 2006]. Por exemplo, ela é crucial: (i) para o desenvolvimento de projetos científicos de grande escala cujos dados são produzidos independentemente pelos pesquisadores; (ii) para instituições de pesquisa que precisam produzir relatórios institucionais a partir das publicações de seus pesquisadores que estão espalhadas em uma variedade de fontes, tais como as bases de publicações DBLP e ISI Web of Knowledge, os Currículos Lattes desses pesquisadores e os sistemas corporativos da instituição; e (iii) para o desenvolvimento de portais Web que integram dados de fontes pré-existentes.

No nível de esquema, a necessidade de se especificar correspondências entre esquemas de fontes de dados heterogêneas que se referem a uma mesma entidade surge devido à não uniformização desses esquemas. Por exemplo, uma fonte pode tratar entidades *pessoa* de forma genérica, enquanto que outra fonte pode considerar diferentes tipos de pessoa, como *estudante* e *docente*. Além disto, esquemas de fontes heterogêneas podem possuir nomes de atributos distintos para representar um mesmo conceito. Por exemplo, uma fonte pode armazenar a data em atributos com nome *data*, enquanto outra fonte pode armazenar a mesma informação com o nome *timestamp*. Outro conflito em nível de esquema refere-se ao fato de que atributos que representam o mesmo conceito podem estar armazenados em diferentes tipos de dados. Por exemplo, em uma fonte a data pode ser armazenada no formato *mês/ano*, enquanto que em outra fonte a data pode ser armazenada apenas como *ano*. Portanto, é necessário primeiro integrar os esquemas de fontes heterogêneas para que depois elas possam ser comparadas e integradas em nível de instância.

No nível de instância, a ambiguidade na identificação de entidades surge porque entidades do mundo real são usualmente identificadas por valores de seus atributos, ao invés de por identificadores únicos. O problema consiste, portanto, em determinar quais elementos de fontes distintas referem-se à mesma entidade. Por exemplo, para se integrar publicações, é necessário identificar que a entidade e_1 = ({“Distributed query processing in a relational database system”}, {“169-180”}, {“Robert S. Epstein”;

"Michael Stonebraker"; "Eugene Wong"}) de uma fonte refere-se à entidade e_2 = ({"Distrited query processing in a relational database system"}, {"169-179"},{"Epstein, R.S."; "Stonebraker, M."; "Wong, E."}) de uma outra fonte. A pesquisa sobre ambiguidade na identificação de entidades é bastante extensa, e tem sido atualmente denominada como resolução de entidades e reconciliação de referências [Dorneles et al. 2004, Chen et al. 2005, Dong et al. 2005, Gal et al. 2005, Aleman-Meza et al. 2006, Kalashnikov and Mehrotra 2006, Karger and Jones 2006, On et al. 2006, Bhattacharya and Getoor 2007, Chen et al. 2007, Dorneles et al. 2007, Borges et al. 2008, Benjelloun et al. 2009]. Basicamente, as técnicas existentes na literatura visam a geração de agrupamentos de entidades que têm certo grau de similaridade entre si e que, portanto, têm alta probabilidade de serem a mesma entidade do mundo real.

Ainda com relação ao nível de instância, o conflito de valores de atributos refere-se ao fato de que diferentes fontes podem possuir valores conflitantes para atributos de entidades similares. No exemplo anterior, os valores dos atributos {"Distributed query processing in a relational database system"} de e_1 e {"Distrited query processing in a relational database system"} de e_2, bem como os valores dos atributos {"169-180"} de e_1 e {"169-179"} de e_2 possuem valores conflitantes. Para cada agrupamento pode ser útil gerar uma entidade integrada que represente todas as entidades similares, contendo apenas dados integrados que sejam os mais corretos. Portanto, o valor de cada atributo da entidade integrada deve ser escolhido para ser um dos valores encontrados nas entidades similares, para resolver o conflito de valores de atributos.

2.2. Procedência dos Dados

Quatro aspectos devem ser considerados para o desenvolvimento de modelos de procedência dos dados. O primeiro deles enfoca "quais dados de procedência armazenar", e inclui a definição de quais tipos de dados de procedência devem ser armazenados e qual a granularidade desses dados. Com relação aos tipos de dados, a procedência recebe diferentes classificações na literatura, como *source* e *transformation provenance* [Glavic and Ditt 2007], *process* e *provenance meta-information* [Del Rio and Silva 2007], *prospective* e *retrospective-provenance* [Zhao et al. 2006], *where* e *why-provenance* [Buneman et al. 2001] e *when-how-*

what [Widom 2005]. De forma mais específica ou mais genérica, essas classificações referem-se às informações sobre as fontes de dados e sobre os processos de transformação pelos quais os dados foram submetidos. Com relação à granularidade, os dados sobre a procedência podem ser coletados em diversos níveis de detalhe. Em um banco de dados relacional, por exemplo, eles podem ser armazenados em nível de tabela (maior granularidade), tupla (média granularidade) ou atributo (menor granularidade) [Glavic and Ditt 2007]. Além disso, os dados sobre procedência podem ser vistos como resultados de operações, as quais podem ter diferentes tipos de dados e granularidades. A definição das operações é específica de cada aplicação, tal como descrito na seção 3.1.

Após definir os tipos de dados a serem armazenados, deve-se estabelecer uma estratégia para "como coletar os dados de procedência". Essa coleta pode ser feita com a intervenção do usuário [Buneman et al. 2006b] ou automaticamente [Munroe et al. 2006, Muniswamy-Reddy et al. 2006, Del Rio and Silva 2007, Archer et al. 2009]. Quanto ao momento em que a coleta é feita, a abordagem *lazy* coleta a procedência de um dado apenas quando ela é requisitada, enquanto a abordagem *eager* armazena a procedência conforme o dado passa de um sistema para outro ou sofre transformações [Tan 2004].

Na sequência, é necessário "armazenar os dados de procedência" para torná-los acessíveis futuramente. A procedência pode ser armazenada junto com o dado ao qual ela se refere [Widom 2005], ou separadamente [Zhao et al. 2006, Buneman et al. 2006a]. Além disso, desde que dados de procedência são volumosos, eles devem ser estruturados visando diminuir o espaço de armazenamento [Buneman et al. 2006a, Chapman et al. 2008, Heinis and Alonso 2008, Anand et al. 2009].

Um último desafio é tornar os dados de procedência disponíveis para que os usuários possam consultá-los, ou seja, definir "como consultar os dados armazenados". Consultas do tipo rastreamento consistem em consultar os dados e verificar a procedência dos mesmos (e.g., Como o relatório foi gerado?). Consultas do tipo filtro consistem em consultar os dados filtrando-os por algum critério de procedência (e.g., Gerar um relatório com dados advindos apenas de Currículos Lattes).