Attribution Principles for Data Integration: Technology Perspectives

February 2002

Thomas Lee

Department of Operations and Information Management
University of Pennsylvania, The Wharton School

This page left intentionally blank.

Attribution Principles for Data Integration: Technology Perspectives

by
Thomas Y. Lee

Abstract
This paper is excerpted from a thesis submitted to the Engineering Systems Division in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the Massachusetts Institute of Technology in January of 2002. This paper addresses problems of attribution that arise from the data integration that is exemplified by data re-use and re-distribution on the Web. We present two different perspectives. We begin with a simple definition of attribution, asking *what* data are we interested in and *where* does it come from? A formal model and its properties are defined, implementation in an extended relational algebra is described, and application to semistructured data on the Web is discussed.

Our formal model of attribution is developed in the established foundation of the Domain Relational Calculus (DRC). Three distinct types of attribution are identified: comprehensive, source, and relevant. For each type, we consider the attribution of equivalent DRC expressions, attribution for composed queries, and granularity. An algebra is presented to implement the model. The extended algebra is closed, reduces to the standard relational algebra, and is a consistent extension of the standard algebra.

Thesis Supervisor: Stuart E. Madnick
Title: John Norris Maguire Professor of Information Technology
MIT Sloan School of Management

1 Introduction

In the legend of Theseus, the hero of Athens entered the Labyrinth of Daedalus on the Isle of Crete to face the Minotaur. Critical to both his successful hunt and victorious return was the simple ball of thread that Theseus used to trace his path. (Bulfinch 2001; Lindemans 2000) As the wealth of content available via electronic networks continues to grow, the Internet has become a maze to rival Daedalus' Labyrinth.

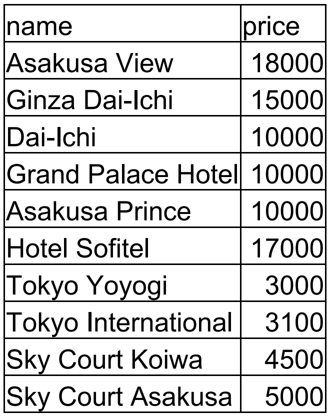
Today, the World Wide Web is a popular way to access the Internet. One group of tools to help people navigate the labyrinth of on-line content are integration services that allow a user to pose rich queries across multiple sites and aggregation services which effectively roll several different sources behind a single point of entry (like Web portals). Consider for example, the case of planning a vacation. The Web may be like having the library on your desktop, but in at least one way, the virtual is no better than the physical. You still must go to the travel section (in the library or on some Web portal like Yahoo!™) and search the different travel guides.

Suppose that you are planning a trip to Japan. There are dozens of on-line resources, many accessible over the Web, ranging from guides for budget conscious travelers (Lonely Planet, Hostelling International) to more traditional guides (Frommer's Travel Guides) to application specific resources (Hotelguide.com, roomz.com). Note that these are resources for researching your trip. We are not discussing transactions such as making reservations or purchasing event tickets.

Rather than laboriously surfing through multiple guides, suppose that you had access to a Travel Resource Integrator (TRI). You might then want to ask:

Q1 What places in Tokyo, Japan may a person traveling alone find a single bed for less than 25,000¥?

The TRI might provide you with the following table:



name	price
Asakusa View	18000
Ginza Dai-Ichi	15000
Dai-Ichi	10000
Grand Palace Hotel	10000
Asakusa Prince	10000
Hotel Sofitel	17000
Tokyo Yoyogi	3000
Tokyo International	3100
Sky Court Koiwa	4500
Sky Court Asakusa	5000

Table 1.1 Results for Q1

While demonstrating the convenience of such a tool, this example also serves to illustrate at least one specific problem with data integration tools like the TRI that applies not only to users but to providers of on-line resources such as those accessible over the Web. Specifically,

Where does this information come from?

You as a user might like to know where the information comes from for reasons such as quality or search. Some questions related to quality that you might wonder include:

- Do you trust the source of this hotel list?
- Does this hotel list draw upon established, reputable resources such as Frommer's or Baedeker's, or is the list compiled from the memories of people who traveled to Tokyo twenty years ago?
- Is the information in the list current? Hotel prices often fluctuate significantly depending upon the time of year you wish to travel. Are all of the listed establishments still in business?

Even if you assumed the veracity of the content, once you had a list, you might want to read more about a specific hotel. To read additional information, you would want to look in the guide where you originally learned about the hotel in question. For example, you would want to know that the listing for the *Asakusa View* came from the Frommer's. Additional information that might be answered from the sources include:

- Are any on this list single beds (e.g. youth hostels) rather than single rooms?
- Which of these lodging options, if any, are located by interesting tourist attractions?
- How can I make a reservation at one of these listings? Is there a phone number to call?

Information providers also have an interest in knowing where information comes from and how data flows. Who should receive acknowledgement for preparing the data in your query result? Who should be paid for this data? If the information is older than the copyright term limit, is the content transferred to the public domain (and therefore free). However, how would individual users know which data fit that category? A single query, moreover, may use information from more than one place. How are rights and remuneration rationed between different contributors? The problem, for both users and the market as a whole, made difficult by the migration from physical to electronic, is only exacerbated by the Web, which makes it easy for people to link and frame or copy content from other sources.

In summary, then, we have suggested three general reasons why attribution is important: data quality, search, and intellectual property.

The question of attribution and its implications is not merely speculative. mySimon Inc. is a comparison shopping service that aggregates data from a number of on-line catalogs in a single data warehouse to facilitate user search. In 1999, mySimon brought suit against Priceman, another comparison shopping service, charging, among other claims, that "Priceman did not sufficiently attribute its meta-search results to mySimon (Kaplan 1999)."

eBay, Inc. hosts an on-line auction house that allows users to play the parts of both buyer and seller. Sellers post items for auction in a database of products that buyers may browse or search and bid for. Bidder's Edge (BE), a comparison service not unlike mySimon or Priceman, warehoused the contents of several auction houses including eBay, Amazon, and Yahoo. eBay won a preliminary injunction against BE's practice in a lawsuit that included the complaint that "caching can lead to outdated information ... potentially harming eBay's reputation (Krebs 2000)."

While these two cases highlight the relevance of attribution-related issues, they also highlight a third point, the legal distinction between individual users and third party services. Suppose that eBay and mySimon were on-line travel resources. An individual user, like a physical shopper, could certainly have behaved like an integrator by visiting different stores and comparing prices without inducing any lawsuits. What if you asked a friend to shop for you, however? What if you paid a personal assistant to shop on your behalf? What about a commercial service? Finally, to what degree can the integration service "anticipate" your requests and search in advance? Ultimately, how far removed from an individual user can an integration service stray while still claiming to "stand in the shoes" of that user?

Details of these cases and others are discussed in a separate paper on policy perspectives to the attribution problem space. However, even this brief introduction serves to illustrate the tension generated by integration: Users benefit from integration, but integration can reduce a database producer's incentives to the point that there are no databases to integrate. As Senator DeWine explained, the threat is that "investment in databases will diminish over time.... Ultimately, the reliability of information available to consumers over the Internet would be undermined (MacMillan 2000)."

The thesis from which this paper is excerpted is about technologies and policies for balancing the tension between database integration and database production. Data integration is a challenging problem with issues that range from the technical (e.g. semantic and syntactic heterogeneity between sources (Goh 1997; Wiederhold 1992) to policy (e.g. standards for data organization and presentation (e.g. EDI, ASN.1, XML). This thesis identifies a set of challenges to integration that stem from the problem of attribution (i.e. knowing where data comes from). The challenges embrace a range of technology and policy questions. Therefore, the thesis is divided into two parts. In this thesis excerpt, which constitutes Part 1, we adopt a technology-based approach to documenting data sources. A formal model of attribution is introduced to support the capability of integrating data from heterogeneous sources. In a separate thesis excerpt, Part 2, we expand the scope of our examination from technologies that support data integration to the general issue of data integration regardless of the means for doing so. Policy measures to both limit and support integration based upon where information comes from are considered.

In Part 1, we propose one technological approach to addressing attribution-related challenges. We develop a formal model of attribution in the context of the relational data model. Although motivation for this work largely stems from efforts to introduce transparency to the heterogeneous, semistructured environment that is the World Wide Web, we build our theory in the relational context because the relational data model provides firm theoretical grounding and is the foundation for the most widely used commercial database products today.

Because of society's ever deepening dependence upon streams of data, we have not been the only individuals interested in the integration-attribution problem space. It becomes clear that over time, no small amount of theoretical and empirical research, often in different guises, has already been leveled at the general problem of attribution. Section 2 provides a very brief overview of a number of the diverse, perhaps seemingly unrelated research streams. Research approaches and results more similar to our own or upon which we draw heavily are revisited and discussed in greater detail throughout the thesis.

Section 3 provides a high-level tour of the model. Through examples and illustrations, we attempt to provide an intuition for the different concepts and principles that the model aims to characterize. In Section 4, we extend our intuitions to a formal model. Our goal in providing a formal model is to offer a consistent framework for interpreting different facets of attribution and understanding how those different dimensions relate to one another. Our formalization is based upon the proof semantics of the domain relational calculus (DRC). A brief review of the specific syntax and semantics assumed is provided.

After presenting the model and some of its properties, we extend the relational algebra in Section 5 to support one instance of the model. We consider some general properties of algebraic extensions such as closure and expressiveness. Then we evaluate the degree to which the extended algebra implements the model. Finally, revisiting the example from Section 6 that originally motivated our exploration of attribution, we begin a discussion of extensions to our model of attribution.

2 Related work

As evidenced by the history of research in citations and references, attribution existed as a general principle of data management long before the advent of digital media and electronic databases (IFLA 2002). The need for attribution is only exacerbated by the medium for widespread data reuse and redistribution that defines the World Wide Web. Therefore, it is perhaps not surprising that there is a great deal of research that relates in one measure or another to the attribution problem space as articulated in Section 1.

Rather than attempting to survey the entire body of related work, we focus on research most similar to our own. Where useful to do so, we attempt to direct the reader to specific application domains or other lines of work that may prove fruitful either for future extensions or to complement that which is presented in this thesis.

We defined the breadth of the problem space in Section 1 based upon the dimensions of *who* is gathering and integrating data, *what* data is gathered, *where* the data comes from, *when* the data is collected, *why* or on whose behalf the content is collected, and *how* the integrated collection is used. While there are many technology-based approaches to specific dimensions of the problem (e.g. cookies and Web logs are two approaches to identifying *who*), attribution focuses on drawing the connection between *what* and *where*.

2.1 Formal approaches

Research on the relationship between *what* and *where* falls is separable into formal approaches and pragmatic experience. Pragmatic experience is discussed below. Formal approaches in the literature define attribution in one of two ways: the relational algebra and the relational calculus.

The attribution model developed in this thesis was inspired by the Polygen data model, which was first presented in (Wang and Madnick 1990). Though they do not offer a formal definition, Wang and Madnick implicitly define attribution algebraically, as part of a system to assess data quality in heterogeneous data integration. In a Polygen relation, every value has two sets of metadata associated with it. For each result value, input relations are classified into one of three categories: a *source*, an *intermediate*, or irrelevant. The *source* set and *intermediate* set each constitute a heuristic for assessing the quality of a value and the quality of the overall query result. The *sources* for a value in the result are inductively defined as the algebraic input relations that contain those tuples from which said value derives. *Intermediates* are those relations used to evaluate algebraic selection conditions for the query result. Granularity is introduced implicitly. Specific values in the result (fine-grained result granules) are linked to base relations (coarse-grained source granules).

Sadri's work on Information Source Vectors (ISVs) also provides an implicit, algebraic definition of attribution by defining the quality of a tuple in the query result (Sadri 1991; 1994; 1995). Like the Polygen data model, ISVs also classify input relations into one of three roles. ISVs, however distinguish between corroborating and contradictory sources. A source vector, with one slot for every input relation in the database, is associated with every tuple of every base and intermediate relation. The ISV for a result tuple is inductively derived from the ISVs of the algebraic query inputs. Each source vector implicitly corresponds to our notion of *comprehensive* attribution. Because sources are not distinguished from intermediates, Sadri can associate a source vector with every tuple in a relation rather than every value in a relation.

It is worth noting that there exists a host of other works, some of which we will mention in the context of pragmatic approaches to attribution below, that also rely upon implicit, algebraic definitions of attribution. Domain and application specific research in the area of Census data tracking, Geographic Information Systems, and security authorization (Ferber 1991; 1992; Lanter 1991; Lanter and Surbey 1994; Motro 1996; Motro and Rakov 1998; Rosenthal and Sciore 1999a; b; Woodruff and Stonebraker 1997) all determine some meta-characteristic of a value or a tuple in a result based upon the processing of input relations. Some (Woodruff and Stonebraker 1997) define fine-grained lineage, associating result values with input values rather than input relations. Note that we may frame some of the research in probabilistic or temporal databases similarly (Dey, Barron, and Storey 1996; Dey and Sarkar 1996). The probabilities or temporal ranges are a function of the constituent inputs. From the perspective of defining attribution based upon the query processing operations, however, they are all essentially similar.

The research in this thesis builds from earlier work that combines the concept of attribution with a specific metric that derives from the input relations such as data quality or access permissions. We extend the existing literature in several respects. First, we provide an explicit definition of attribution. This definition is couched in terms of the relational calculus and the logical foundations for relational database theory rather than implicitly in the algebra. Second, we refine the concepts of *source* and *intermediate* to distinguish between three types of attribution, *comprehensive*, *source*, and *relevant,* to correspond to different user needs. Third, based upon the formal model we can express equivalence properties for attribution. Finally, we attempt to articulate granularities explicitly and then suggest how the relationship between source and result granules may support subsequent algebraic extensions to reduce the burden of propagating attribution metadata.

In contrast to the implicit algebraic definitions of some of the early work in source tracking, Cui et al. (2000; 2001; 1997 (revised 1999)) provides a formal definition of *lineage*, in terms of the relational algebra. Reflecting their primary application domain, data warehousing, Cui et al. further extend their definition of lineage first to encompass bag semantics and aggregation functions and later to more general classes of transformations (e.g. arithmetic functions in a select clause, grouping tuples, etc.). For the base relational operators, the lineage of a result is recursively defined by the successive application of operators in the query tree. Equivalence properties of lineage are defined. As with Sadri (1991), corresponding to their focus on comprehensive attribution, Cui et al. (1997 (revised 1999)) define attribution for result tuples. Unlike earlier work, however, they focus on "fine-grained" lineage and associate result tuples with input source tuples rather than input relations.

Given our characterization of the attribution problem space, we define three different types of attribution rather than one. Each type of attribution has somewhat different properties with respect to both equivalence and granularity. *Lineage*, as defined in (Cui, Widom, and Wiener 1997 (revised 1999)), corresponds to our concept of *comprehensive* attribution. We also attempt to define the relationship between source and result granules explicitly.

The relational calculus and the relational algebra are equal in their expressiveness. Consequently, neither model is necessarily better than the other for defining attribution. However, as is echoed in the work by Buneman et al. (1998; 2001), the different semantics of calculus queries provides a more direct parallel to languages for querying semistructured data on the Web; and it is the reuse and redistribution exacerbated by the Web that underlies our interest in attribution.

The second category of theoretical approaches builds or borrows from the first-order predicate logic with which the relational calculus is defined. In the relational calculus, queries take the form of expressions on predicates that represent relations. Intuitively, values in a query result are attributable to values from the relational predicates that make the query expression true.

Panorama (Motro 1996) is a system for assessing the quality of data in a query result. Panorama explicitly notes that the same quality assessment(s) might not apply uniformly to all values in the relation (*granularity*). The reliability or completeness of answers are at least partially determined by their contributing sources. Quality properties are thus associated with the subset of tuples in a relation for which the property holds. A tuple subset is proscribed by a *meta-tuple* or select-project view expressed in the relational calculus. A particular property is inherited by a query result if tuples from the corresponding *meta-tuple* provide a true interpretation of the query expression.

Using query expressions to define *meta-tuples* matches our use of expressions to define *source granules*. We extend the intuition one step further to associate source granules with result values rather than tuples. This finer granularity supports three different types of attribution. By contrast, Panorama propagates values based upon our notion of *source attribution* or the specific *meta-tuple(s)* or relations from which result tuples are drawn. Finally, we do not associate source granules with particular properties of the sources, thereby separating the attribution from a specific motivation (e.g. quality, intellectual property, search), leaving the user or application domain to associate their own meta-characteristics.

Buneman et al. (1998; 2001) borrow from the logical intuitions underlying the relational calculus, but generalize the data model to a deterministic semistructured data model. They define both *why* and *where* data provenance for queries (path expressions) in this context. In a separate work, Buneman et al. (2001; 2001) represents the concept of source granules as deep linking into source documents. They also explore the use of key values (in the relational sense) to represent linking into source documents.

The research by Buneman et al. is in many ways most similar to the spirit, approach, and ultimate direction that we aim to pursue in this thesis. Indeed although we structure our formal model in the relational framework to leverage existing results, our initial motivation and long-term aim all along has been to extend the model to semistructured data on the Web. Many of our early intuitions about attribution, such as attribution composition or source and result granularity, stem from this semistructured orientation (Lee, Bressan, and Madnick 1997; 1998).

The semistructured data model is more general than the relational model from which we build in this thesis. However, using the terminology loosely, the *why* provenance for a query on semistructured data is the set of sub-trees that matches the path expression in the same way that we define *comprehensive* attribution as the set of substitutions that provides a true interpretation of a calculus query expression. Indeed (Buneman, Khanna, and Tan 2001) draws upon the same conjunctive query literature that we leverage in exploring equivalence properties (Klug 1988; Sagiv and Yannakakis 1980; Ullman 1989). Similarly, *where* provenance corresponds to our notion of *source* attribution, which in turn stems from the *source* set for every value in a Polygen relation.

Framing our work in the relational calculus, as noted earlier, allows us to borrow directly from the existing literature on equivalence and containment. We are, however, limited to intuitions and observations about the parallels to querying in semistructured environments. We introduce three types of attribution, which better support not only the motivations of the attribution problem space but relate to the relationship between source and result granules. We also treat explicit equality in theta comparisons independently of the natural join. This reflects a distinction in *source* attribution (*where* provenance) relevant to such purposes as intellectual property or remuneration. The natural join suggests that both relations are sources for the join attribute whereas explicit equality indicates that each argument to the equality has only one, distinct source. Finally, we also present an extension to the relational algebra as a mechanism for explicitly propagating attribution metadata in annotations.

2.2 Pragmatic approaches

Turning from different formal methods for defining attribution, we next consider pragmatic approaches to providing attribution support in querying and integration.
We can separate pragmatic strategies for managing attribution into eager and lazy approaches. Eager approaches continuously update and propagate attribution metadata as a part of query processing. A'priori evaluation, however, amortizes the cost of attribution maintenance over multiple values in the data set and minimizes response time to requests for attribution. We may also think of eager approaches as bottom-up approaches that recursively maintain attribution values.

By contrast, lazy approaches, which may also be thought of as top-down approaches, begin with a query result and drill backwards to trace sources for specific values only in response to specific requests. Minimal expense is incurred in query processing, but the cost of responding to any single attribution request is much higher. Hybrid models may evaluate the attribution for certain intermediate inputs (e.g. frequently used views) to speed-up response to ex-post, lazy attribution requests.

Early work on extensions to the relational data model were, in part, both motivated by and demonstrated using eager attribution principles. Schek and Pistor (1982) articulated their approach to the non-first normal form in the context of merging information retrieval and database approaches to managing search. In their NF2 model, data values are extended with a relation identifying their source(s) as a means for directing subsequent information retrieval queries for additional data. Their early work echoes an attribution driver identified in Section 1, searching for related information.

The Polygen data model (Wang and Madnick 1990), upon which this thesis is based, is another prototypical example of an eager approach to attribution. Wang and Madnick extend the relational data model with two annotations - one each for references to *sources* and references to *intermediates*. Every domain value is therefore a triple and a relation is a finite subset of the Cartesian product of such triples. Polygen extensions to the algebra then update values in the *source* and *intermediate* annotations with each successive application of the corresponding operator. References are relation names. The Polygen model therefore provides attribution for individual result values using relation-level source granules.

A number of projects that calculate and propagate meta-attributes of data (e.g. time stamps, probability, quality, authorization) work in a similar manner. In (Dey, Barron, and Storey 1996; Dey and Sarkar 1996), a tuple is tagged with a probability measure or time stamp, respectively. The preservation of certain algebraic equivalencies is demonstrated and, in the case of the temporal relational algebra, aggregation functions are also considered. Both closure and consistency with the traditional relational algebra are verified. Tuples are tagged similarly with quality specifications in (Motro and Rakov 1998). Algebraic extensions manage metadata propagation from constituent inputs to results. In (Rosenthal and Sciore 1999b), security policies are specified as the manner by which security authorizations are aggregated. For example, the permissions on a specific tuple might be the least upper bound of the permissions on all inputs.

That different projects may calculate meta-characteristics at different levels of granularity is perhaps more a function of the application domain than a limitation of the eager approach. Certain applications (e.g. intelletctual property), may wish to identify the Source of a specific value in a tuple while other uses of attribution may require only tuple-level granularity. The principle distinction between these domain specific approaches and the work in this thesis (as well as the Polygen data model from which this work derives) is the propagation of source meta-characteristics (e.g. quality) rather than source references.

Sadri's (1991; 1994; 1995) work on Information Source Vectors (ISVs) suggests the complementary nature of the two approaches to annotation. The relational data model is extended with an ISV annotation for every tuple. Algebraic extensions update and propagate ISVs for result tuples. The quality of a given tuple is then determined as a function of the corroborating and contradictory sources in the corresponding ISV rather than returning a continuously updated metacharacteristic. Where ISVs are associated with result tuples, the attribution in this work is associated with individual values, thereby supporting distinctions between types of attribution.

In addition to eager approaches that extend the data representation with annotations are eager systems that construct parallel data structures for managing attribution metadata. Panorama is one such system (Motro 1996). In Panorama, annotations on the quality (e.g. soundness, completeness) of tuples in a relation are associated with a *meta-tuple* for the relation. A *meta-tuple* is simply a select-project view defining the subset of tuples to which the metric applies. The set of all metrics applicable to a relation is called a *meta-relation*. Queries on relations are paralleled by operations on the corresponding *meta-relation*.

Where eager approaches propagate data continuously, lazy approaches minimize the ex-ante cost of maintaining attribution. A minimum amount of information is stored. Only when a specific request is initiated, is the attribution for a result calculated.

In his work to support data integration and reuse in Geographic Information Systems (GISs), Lanter maintains GIS metacharacteristics in a parallel data structure (Lanter 1991; Lanter and Surbey 1994). Where algebraic operators in the relational model process relational tuples, GISs process *layers*. Lanter defines a frame-based representation to capture layer-level metacharacteristics including data transformations. Operations on layers are paralleled by the updates to the corresponding knowledge-base tracking GIS processing. Specific metacharacteristics are therefore associated with each layer in the manner of tuple-level result granules. The lineage for a result is generated by tracing backwards through the frames associated with each successive processing step.

Like Lanter's system for Geographic Information Systems, Woodruff and Stonebraker (1997) define a system to trace data lineage. Unlike Lanter's layer-granularity that documents metacharacteristics at the level of a data set, Woodruff and Stonebraker register data transformations and their inverses. The inverses allow users to regenerate specific base level data inputs to the transformation process. Original data values are calculated iteratively by unfolding successive operations. The result is fine-grained lineage that traces from a value in the result to the source input values rather than merely linking result sets to their constituent inputs.

Cui et al. (2001) investigates lineage for general data transformations in the spirit of (Woodruff and Stonebraker 1997). However, it is their earlier work tracing relational queries, described in (Cui and Widom 2000; Cui, Widom, and Wiener 1997 (revised 1999)), that our extended algebra is most similar to. Assuming a canonical form of an algebraic query tree, Cui and Widom algorithmically construct a tracing query that, for a given result tuple, returns the input tuples. The algorithm works by essentially projecting the result tuple as query constraints down the algebraic query. The resulting *lineage* is transitive over intermediate results and through querying on views.

Although the technique does not strictly require maintaining meta information, as used in eager approaches, it is possible to achieve greater efficiency in lazy attribution processing by utilizing eager approaches in a limited manner. Cui et al. (1997 (revised 1999)) discover significant improvement in lazy performance by storing auxiliary views, which we might equate with eager evaluation of attribution metadata for intermediate query results. Maintaining a minimal amount of metadata with query processing also enables Cui et al. to trace backwards through aggregation functions.

We adopt an annotation approach to managing attribution metadata. Based upon our formal definition of attribution and our articulation of granularity, we redefine the extended relational operators to support the formal definition of attribution. Unlike some of the approaches that extend the relational model, we show how general properties of the algebra, such as closure, are preserved. Moreover, unlike approaches that rely upon implicit definitions, we show how the algebraic extensions indeed support our logical intuitions about the different interpretations of attribution. Although the algebra tracks source granules at the granularity of relation names, it is a straightforward extension to consider variable granularity using expressions as in Panorama (Motro 1996) rather than relations (Sadri 1991) or explicit source tuples (Cui, Widom, and Wiener 1997 (revised 1999)).

Annotations in a bottom-up manner seems the most general approach for addressing the myriad interests that we initially identified in attribution. Certainly systems designed with specific goals in mind might prefer one particular approach over another. Moreover, the top-down query tracing implemented by Cui et al. is similar in spirit to how Panorama associates result granules with source granules and how we project substitutions onto intermediate relational predicates in attribution composition.

Where *meta-tuples* in Panorama or the metadata in other systems to document data probabilities, quality, or authorization (Dey, Barron, and Storey 1996; Dey and Sarkar 1996; Motro 1996; Motro and Rakov 1998; Rosenthal and Sciore 1999a) are explicitly associated with specific metrics, we define attribution only as the association between source and result granules. Doing so allows us to define different types of attribution and to parametrize attribution with different functions for quality, intellectual property, or search metrics as the need arises.

3 Attribution intuitions

In Section 1, we provided some rough boundaries about the attribution problem space and some desiderata for a formal approach to that space. Here, we begin Part 1 of the thesis. Beginning with Section 3 and extending through Section 6, we develop a model for attribution. Although we make the model formal in Section 4, we begin in this Section by attempting to provide the intuitions behind the features and properties of our proposed model. The intuitions are intended to connect the reader from the problem space defined in Section 1 to the formalisms in Section 4. After presenting the model, we operationalize one instance of the model as an extension to the relational algebra. Finally, we consider how the model might apply in the emerging semi-structured data environment.

Throughout this Section and the remainder of this thesis, we couch many of our examples in the context of the relations listed in Table 3.1. The six relations in Table 3.1 represent a number of separate (Web accessible) data sources concerning lodging and tourist attractions in Tokyo, Japan. The relation hotels(HNAME, ROOM, PRICE) lists hotels in Tokyo along with a minimum price for rooms in the ROOM category. The relation sites(SNAME, REGION) identifies tourist attractions in Tokyo along with the general vicinity where the attraction is located. The three relations roughguides(HNAME, PRICE, STATION, PHONE); jyh(HNAME, PRICE, STATION, PHONE, FAX); and hostels(HNAME, PRICE, STATION) all provide listings of youth hostels or other low-budget lodging in Tokyo. The attribute STATION identifies the closest rail station to the associated lodging. regions(HNAME, REGION) provides the general geographic location of selected Tokyo hotels. Though the model is developed in the DRC, for readability, the examples in this Section are posed in English, SQL, and the calculus.

3.1 The meaning of attribution

This theory of attribution is based upon the domain relational calculus (DRC), a logical formalism for representing and evaluating relations between data domains. We build our model in this environment because, while our motivation is heavily influenced by the rapid evolution of data integration on the World Wide Web, most of what is known today about managing and manipulating data is rooted in relational terms. The calculus is also the foundation for SQL, one of the most widely recognized and used standards for querying and managing information. In theoretical terms, then, the calculus will allow us to be precise about our observations and intuitions. Pragmatically, much of the data being used today, even that accessible over the Web, is still managed and manipulated using relational tools built on the calculus.

hotels

HNAME	ROOM	PRICE
Asakusa View	single	18000
Asakusa View	double	20000
Ginza Dai-Ichi	single	15000
Ginza Dai-Ichi	double	25000
Imperial Hotel	single	34000
Imperial Hotel	double	39000
Dai-Ichi	single	10000
Dai-Ichi	double	80000
Grand Palace Hotel	single	10000
Grand Palace Hotel	double	31000
Asakusa Prince	single	10000
Asakusa Prince	double	42000
Hotel Sofitel	single	17000
Hotel Sofitel	double	22000

sites

SNAME	REGION
Imperial Palace	Hibiya
Tourist Information Center	Hibiya
Tsukiji Fish Market	Hibiya
Hama Rikyu Garden	Tsukiji
Sensoji Temple	Tsukiji
Nakamise Dori	Asakusa
Ameya Yokocho	Asakusa
Ueno Park	Ueno
Tokyo National Museum	Ueno
Yanaka	Ueno
Meji Jingu Shrine	Ueno

roughguides

HNAME	PRICE	STATION	PHONE
Sky Court Asakusa	5000	Asakusa	81-3-3672-4411
Hotel Pine Hill	10000	Ueno-Hirokoji	81-3-3822-2251
Sawanoya Ryoken	5000	Nezu	81-3-3847-4477
Hotel Top Asakusa	7000	Asakusa	81-3-3822-1611
Ryokan Shigetsu	7000	Asakusa	81-3-3843-2345

jyh

HNAME	PRICE	STATION	PHONE	FAX
Tokyo Yoyogi	3000	Sangubashi	81-3-3467-0163	81-3-3467-9417
Tokyo International	3100	Iidabashi	81-3-3235-1107	81-3-3267-4000
Sky Court Koiwa	4500	Koiwa	81-3-3672-4411	81-3-3672-4400
Sky Court Asakusa	5000	Asakusa	81-3-3672-4411	81-3-3875-4941

hostels

HNAME	PRICE	STATION
Tokyo Yoyogi	3000	Sangubashi
Tokyo International	3100	Iidabashi
Sky Court Koiwa	4500	Koiwa
Sky Court Asakusa	5000	Asakusa
Hotel Pine Hill	10000	Ueno-Hirokoji
Sawanoya Ryoken	5000	Nezu
Hotel Top Asakusa	7000	Asakusa
Ryokan Shigetsu	7000	Asakusa

regions

HNAME	REGION
Hotel Sofitel	Ueno
Katsutaro	Ueno
Dai-Ichi Hotel	Hibiya
Imperial Hotel	Hibiya
Asakusa View	Asakusa

Table 3.1 Data for examples

The interpretation of a calculus expression is the set of variable substitutions that correspond to facts in the database and make the formula of the expression true (Maier 1983). In the most general sense, we express attribution in terms of the substitutions that make the interpretation of the expression true.

Example 3.1 Intuition for attribution

Q1. Based upon the database of Table 3.1, we might ask: What are the names of all known lodging establishments in Tokyo, Japan? We could answer this question by considering the union of a query on the relation hotels and a query on the relation hostels.

```
SQL 1.1   select HNAME from hotels
          union
          select HNAME from hostels
```

DRC 1.1 {HNAME | hotels(HNAME, ROOMS, PRICE) ∨ hostels(HNAME, PRICE, STATION)}

The query result is:

HNAME
Tokyo Yoyogi
Tokyo International
Sky Court Koiwa
Sky Court Asakusa
Hotel Pine Hill
Sawanoya Ryoken
Hotel Top Asakusa
Ryokan Shigetsu
Asakusa View
Ginza Dai-Ichi
Imperial Hotel
Dai-Ichi
Grand Palace Hotel
Asakusa Prince
Hotel Sofitel

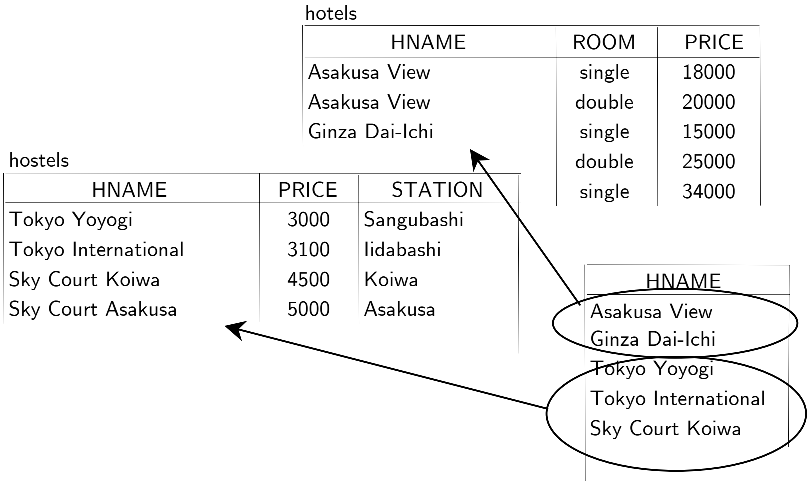
Table 3.2 Lodging establishments in Tokyo, Japan

Some of the substitutions that provide true interpretations include the following:
<*f*("Asakusa View"/HNAME, "single"/ROOMS, 18000/PRICE)>;
<*g*("Tokyo Yoyogi"/HNAME, 3000/PRICE, "Sangubashi"/STATION)>;
<*g*("Sky Court Asakusa"/HNAME, 5000/PRICE, "Asakusa"/STATION)>;
<*f*("Dai-Ichi"/HNAME, "double"/ROOMS, 10000/PRICE)> □

If we further represent relations as *sources* for data, we can talk about different roles that sources play based upon the substitutions (facts) from each source used to interpret the expression. Future references to '*sources*' in this Section will refer to the relations containing the facts which, when substituted into the query expression, produce a true interpretation.

Example 3.2 Intuition for a "source"

Given the substitutions for Q1 in Example 3.1, the corresponding *sources* are: relation hotels and relation hostels. We depict this intuition in Figure 3.1. From the answer, a list of HNAME, we can trace backwards to the corresponding input relations. □



hotels

HNAME	ROOM	PRICE
Asakusa View	single	18000
Asakusa View	double	20000
Ginza Dai-Ichi	single	15000
	double	25000
	single	34000

hostels

HNAME	PRICE	STATION
Tokyo Yoyogi	3000	Sangubashi
Tokyo International	3100	Iidabashi
Sky Court Koiwa	4500	Koiwa
Sky Court Asakusa	5000	Asakusa

HNAME
Asakusa View
Ginza Dai-Ichi
Tokyo Yoyogi
Tokyo International
Sky Court Koiwa

Figure 3.1 Intuition for a "source"

We saw in Section 1 that there may be different motivations for or interests in attribution. Accordingly, our theory defines three explicit types of attribution: comprehensive, source only, and relevant. *Comprehensive* attribution identifies everything that was used to evaluate an expression. It identifies every source that was consulted. Certainly from the perspective of remuneration, comprehensive attribution is in the interests of data providers. From a data quality perspective, comprehensive attribution provides a measure of completeness regarding the answer to a query.

Source attribution, by contrast, recognizes the difference between "supporting material" and the actual facts. *Source* attribution identifies the specific relations from which a query result is drawn. We use the metaphor of a footnote in a text citation. Unlike the *comprehensive* listing of references in a bibliography, a footnote identifies author, title, and page number for a specific fact, figure, or quotation. Certainly for intellectual property purposes, source attribution is critical. Moreover, as measure of quality distinct from that of *comprehensive* attribution, we may use the credibility of a given source to label the veracity of the data from that source. Finally, knowing the specific source of a data item provides us with a starting point for seeking additional, related information.

Relevant attribution constitutes a subset of *comprehensive* attribution. Given a specific result, the *relevant* attribution identifies the subset of *comprehensive* references that are associated with the *source* attribution of a particular query. For example, the *comprehensive* list of references in this thesis numbers over 250 separate works. However, our treatment of negation in Section 4 draws from work by Sagiv and

Yannakakis (Sagiv and Yannakakis 1980). However, we found this reference through a series of other works (Abiteboul, Hull, and Vianu 1995; Ullman 1989). *Relevant* attribution therefore traces the supporting material used to arrive at a single query. In SQL terms, we may think of *relevant* sources as those used in evaluating selection conditions.

In simple queries, the *comprehensive, source,* and *relevant* attribution may look identical. As query complexity increases, however, particularly in the light of the data environment of the Web, such distinctions may become increasingly important in parsing the attribution problem space.

Example 3.3 Types of attribution

Q2. Consider the query where we ask for all hotels by the Imperial Palace in Tokyo, Japan. Based upon the hypothetical database of Table 3.1, we have:

```
SQL 2.1   select HNAME
          from hotels, regions, sites
          where sites.SNAME = "Imperial Palace"
          and sites.REGION = regions.REGION
          and hotels.HNAME = regions.HNAME
```

DRC2.1 {HNAME | regions(HNAME, REGION) ∧ sites("Imperial Palace", REGION) ∧ hotels(HNAME, ROOMS, PRICE)}

The substitutions include (but are not limited to):

<*f*("Imperial"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME, "single"/ROOMS, 34000/PRICE)>;
<*f*("Imperial"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME, "double"/ROOMS, 39000/PRICE) >;
<*f*("Dai-Ichi"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME, "single"/ROOMS, 10000/PRICE) >;
<*f*("Dai-Ichi"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME, "double"/ROOMS, 80000/PRICE) >;

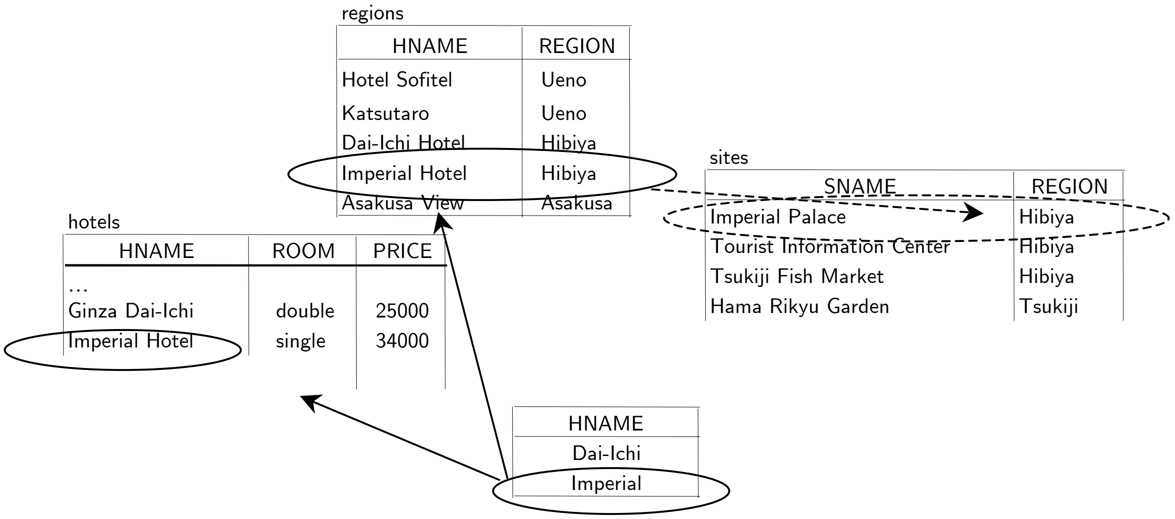
Now, consider the relations from where these substitutions are drawn. The different substitutions are drawn from three different relations. Therefore, the *comprehensive* attribution includes these three relations. But, not all of the relations in the FROM clause of the SQL query are used to provide answers. As illustrated in Figure 3.2, some sources are used to evaluate selection conditions rather than provide selection attributes. In particular, the HNAME attribute that constitutes the query result appears in only two of the queried relations. Thus, the *source* attribution includes only two relation names. Finally, because the relation sites is used in evaluating selection conditions, we include it in the *relevant* attribution.[1]

[1] For an example where *comprehensive*, *source* and *relevant* attribution are all different for the same query expression, see Example 3.7 where we consider the Union query operator.

comprehensive attribution
{<regions; sites; hotels>}

source attribution
{<regions; hotels>}

relevant attribution
{<regions; sites; hotels>} □



HNAME	REGION
Hotel Sofitel	Ueno
Katsutaro	Ueno
Dai-Ichi Hotel	Hibiya
Imperial Hotel	Hibiya
Asakusa View	Asakusa

SNAME	REGION
Imperial Palace	Hibiya
Tourist Information Center	Hibiya
Tsukiji Fish Market	Hibiya
Hama Rikyu Garden	Tsukiji

HNAME	ROOM	PRICE
...		
Ginza Dai-Ichi	double	25000
Imperial Hotel	single	34000

HNAME
Dai-Ichi
Imperial

Figure 3.2 Example of source attribution

3.2 Properties of attribution

A specific challenge to any theory of attribution is treatment of multiple derivations. Data may derive from many different sources and/or diverse combinations of sources. Accordingly, this theory identifies several distinct categories of multiple derivations and provides an explicit treatment for each. We loosely separate multiple derivations into two categories. Case 1 concerns multiple queries that (appear to) achieve the same result. Think of this as asking the same question in two different ways. For example, "What is for dinner" rather than "What are we eating tonight?" Case 2 addresses a single query that may produce the same answer from more than one source. For example, to discover all the hostels in Tokyo, Japan, you might combine the results from looking in both a Japanese travel guide and an international youth hostel guide. Some entries might be listed in both places.

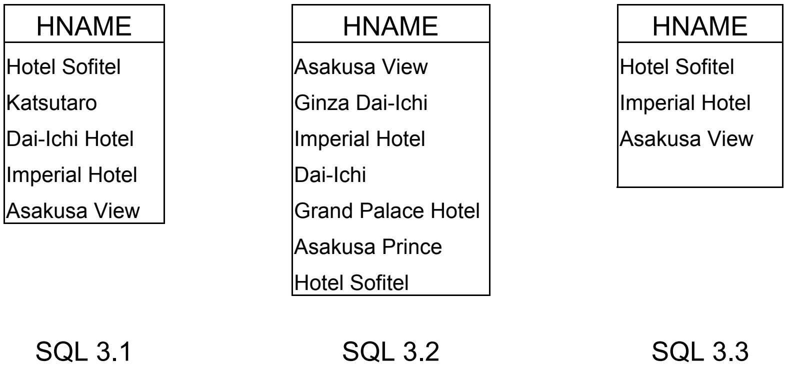
Case 1, multiple queries that (appear) to achieve the same result, is further separated into three classes: weak equivalence, strict equivalence, and composition. Weak equivalence, in a colloquial sense, refers to queries that, perhaps in some circumstances, appear as if they should be equivalent yet are not logically equivalent and therefore vulnerable to incomplete data or other contextual limitations (Ullman 1989).

Example 3.4 Weak equivalence

Q3. Consider the query that asks for all hotels in Tokyo, Japan. Given only the schemas for the relations in Table 3.1, we might conclude that there are at least three different ways to list hotels in Tokyo.

SQL 3.1 select HNAME from regions
SQL 3.2 select HNAME from hotels
SQL 3.3 select HNAME from regions, hotels where hotels.HNAME = regions.HNAME

Unfortunately, as is often the case in real tables, our example data relations are incomplete. There are a number of dangling tuples (Ullman 1989). The incompleteness is especially apparent when we consider the results from each of SQL 3.1 – 3 as noted in Table 3.3. □



HNAME
Hotel Sofitel
Katsutaro
Dai-Ichi Hotel
Imperial Hotel
Asakusa View

SQL 3.1

HNAME
Asakusa View
Ginza Dai-Ichi
Imperial Hotel
Dai-Ichi
Grand Palace Hotel
Asakusa Prince
Hotel Sofitel

SQL 3.2

HNAME
Hotel Sofitel
Imperial Hotel
Asakusa View

SQL 3.3

Table 3.3 Weak equivalence

In principle, it seems only reasonable that the data in a database should be somehow complete and internally consistent. Yet, different tables appear to list different hotels even though they all purport to list hotels in Tokyo, Japan. Though a subject studied in the query optimization literature, we do not consider weak equivalents to constitute multiple derivations and so treat them as distinct queries and say nothing more about them.

Strict equivalence refers to the characteristic that two queries produce the same result given the same database.[2] We introduce the modifier "strict" to emphasize the fact that the multiple queries use the same data sources.

Example 3.5 Strict equivalence

Consider again Q2 which we can express in the DRC as

[2] We refer to the more formal definition of equivalence based upon containment in Section 4 (Ullman 1989).

DRC 2.1 {HNAME | regions(HNAME, REGION) ∧ sites("Imperial Palace", REGION) ∧
hotels(HNAME, ROOMS, PRICE)}
DRC 2.2 {HNAME | regions(HNAME, REGION) ∧ sites("Imperial Palace", REGION) ∧
hotels(HNAME, ROOMS, PRICE) ∧ regions(AHOTEL,AREGION)}

A substitution for DRC2.1 might look like:
<*f*("Dai-Ichi"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME, "single"/ROOMS, 10000/PRICE)>

A substitution for DRC2.2 might look like:
<*f*("Dai-Ichi"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME, "single"/ROOMS, 10000/PRICE, "Asakusa View"/AHOTEL, "Asakusa"/AREGION)>

Note the similarities between the different substitutions. There are more variables in DRC2.2, yet there is a consistency between the substitutions in DRC2.1 and DRC 2.2. Moreover, our intuitions about attribution are the same for both queries.

comprehensive attribution:
{<regions; sites; hotels>}

source attribution
{<regions; hotels>}

relevant attribution
{<regions; sites; hotels>}

In particular, for the case of strict equivalence, none of the data sources is defined in terms of other available sources. □

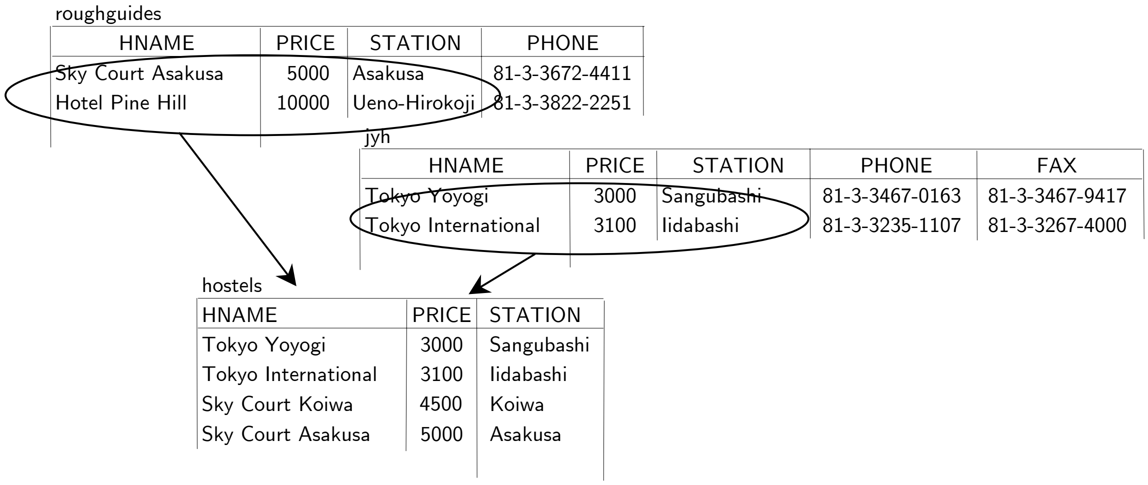
Example 3.6 Defining a source in terms of other sources

Q4. Consider the query for all hostels in Tokyo, Japan
SQL 4.1 select * from hostels

The reliance of multiple intermediaries upon the same underlying base sources is not always immediately apparent, however. For example, we define relation hostels in terms of information from Japan Youth Hostels Association (relation jyh) and Rough Guide Travel (relation rg). The relationship is depicted in Figure 3.3. Data is taken from the constituent relations to construct a new relation.

```
SQL 4.2   select HNAME, PRICE, STATION from jyh
          union
          select HNAME, PRICE, STATION from rg □
```

In focusing only on strict equivalence, we borrow from the query optimization literature to arrive at the result that the attributions for equivalent select, project, join queries involving theta inequality and natural join are, in some sense, the same. Attribution equivalence is evident in Example 3.5 where, although DRC2.2 has more variables and predicates, there is the sense that there is no extra information gained. We make this intuition explicit when we define attribution equivalence more formally in Section 4. However, attribution equivalence is lost for complete and source attribution when we consider queries with union.



roughguides

HNAME	PRICE	STATION	PHONE
Sky Court Asakusa	5000	Asakusa	81-3-3672-4411
Hotel Pine Hill	10000	Ueno-Hirokoji	81-3-3822-2251

jyh

HNAME	PRICE	STATION	PHONE	FAX
Tokyo Yoyogi	3000	Sangubashi	81-3-3467-0163	81-3-3467-9417
Tokyo International	3100	Iidabashi	81-3-3235-1107	81-3-3267-4000

hostels

HNAME	PRICE	STATION
Tokyo Yoyogi	3000	Sangubashi
Tokyo International	3100	Iidabashi
Sky Court Koiwa	4500	Koiwa
Sky Court Asakusa	5000	Asakusa

Figure 3.3 Views: defining sources from other sources

Example 3.7 Attribution equivalence breaks down under union

Consider again Q3, which we defined as all hotels in Tokyo, Japan.

We originally answered this question with

```
SQL 1.1     select HNAME from hotels
            union
            select HNAME from hostels
```

Perversely, we might equally answer the query this way:

```
SQL 1.2     select HNAME from hotels
            union
            select HNAME from hostels
            union
            select HNAME
            from hotels, regions, sites
            where sites.SNAME = "Imperial Palace"
            and sites.REGION = regions.REGION
            and hotels.HNAME = regions.HNAME
```

SQL 1.2 corresponds to:

DRC1.2 {HNAME | hotels(HNAME, ROOMS, PRICE) $\vee$ hostels(HNAME, PRICE, STATION, PHONE) $\vee$ (regions(HNAME, REGION) $\wedge$ sites("Imperial Palace", REGION) $\wedge$ hotels(HNAME, ROOMS, PRICE))}

Compare the attribution between DRC1.1 and DRC1.2 as listed in Table 3.4. In particular, the *source* attribution takes into account the sources used in evaluating each disjunct. However, the third disjunct is arguably irrelevant because any answer in the third disjunct appears also in one of the first two disjuncts.

Returning briefly to Example 3.3, we see that SQL 1.2 and its associated DRC help highlight the intuition behind the different types of attribution. First, consider comprehensive attribution. Each disjunct represents a distinct alternative for satisfying the DRC1.2. However, taken together, every relational predicate in the query expression plays some role in evaluating the result. The reader will note that the query only asks about hotel names (HNAME), however. Hotel names are only found in four of the five relations used in the expression. If, for example we wanted to verify the spelling of a particular hotel name, there would be no reason to return to relation (sites). That relation does not list any hotel names. Source and comprehensive attribution are therefore distinct. Finally, as observed earlier, the third disjunct in DRC1.2 is contained (or subsumed) by the first two disjuncts. As a consequence, the third disjunct cannot impact the query results and so we omit relations from the third disjunct. The third disjunct is not relevant.[3]

□

	DRC1.1	DRC1.2
comprehensive	{<hotels>; <hostels>}	{<hotels>; <hostels>; <hotels; regions; sites>}
source	{<hotels>; <hostels>}	{<hotels>; <hostels>; <hotels; regions>}
relevant	{<hotels>; <hostels>}	{<hotels>; <hostels>}

Table 3.4 Attribution equivalence with union

The "strict" condition contrasts the third class of queries: "composition," where sources are defined in terms of one another. We saw in Example 3.6 what it means for a source to be defined in terms of other sources, often referred to as views.[4] Composition addresses the situation where a query can either be composed on a view or expressed strictly in terms of the original sources underlying any view definition.

[3] It is worth emphasizing that while relations may prove irrelevant, they are not without value. As in comprehensive attribution, we may use equivalent derivation paths to increase our confidence in a particular result. Although outside the scope of this work, we may also consider the role of disjuncts which are, in principle, contained but may contain contradictory information (Sadri 1991).

[4] In the relational context, relations defined in terms of other relations are often referred to as views. In the literature on databases and logic, such relations are referred to as intentional databases or IDB (Ullman 1989).

Example 3.8 Query composition

Q5. Consider a query for all lodging (hostels and hotels) around the Nakamise Dori. Based upon Example 3.6, we know that we can express the query in terms of the relations for hostels and hotels:

```
SQL 5.1   select HNAME
          from hotels, regions, sites
          where hotels.HNAME = regions.HNAME
          and regions.REGION = sites.REGION
          and sites.SNAME = "Nakamise Dori"
          union
          select HNAME
          from hostels, sites
          where hostels.STATION = sites.REGION
          and sites.SNAME = "Nakamise Dori"
```

But, if we know in advance, as we know now, that relation hostels itself gathers information from elsewhere, we can also express the query in terms of the underlying data sources jyh and roughguides (rg) as:

```
SQL 5.2   select HNAME
          from hotels, regions, sites
          where hotels.HNAME = regions.HNAME
          and regions.REGION = sites.REGION
          and sites.SNAME = "Nakamise Dori"
          union
          select HNAME
          from jyh, sites
          where jyh.STATION = sites.REGION
          and sites.SNAME = "Nakamise Dori"
          union
          select HNAME
          from rg, sites
          where rg.STATION = sites.REGION
          and sites.SNAME = "Nakamise Dori"
```

We depict the intuition behind composition in Figure 3.4. SQL 5.1 uses only two relations in the second disjunct (hostels and sites). It is as if relations jyh and roughguides are hidden and inaccessible. Relation hostels then constitutes a view on the underlying sources. The attributions for both queries is shown in Table 3.5. □

	SQL 5.1	SQL 5.2
comprehensive	{<hotels; regions; sites>; <hostels; sites>}	{<hotels; regions; sites>; <jyh; sites>; <rg; sites>}
source	{<hotels; regions>; <hostels>}	{<hotels; regions>; <jyh>; <rg>}
relevant	{<hotels; regions; sites>; <hostels; sites>}	{<hotels; regions; sites>; <jyh; sites>; <rg; sites>}

Table 3.5 Attribution with composed queries

By definition, the query results of equivalent, composed queries are the same. The attributions, however, can be quite different. This seems entirely correct. In the context of distributed, heterogeneous information sources, such as the Web today where data is frequently reused and redistributed, it is not unreasonable to cite an integrator as a source. Factors that are beyond the scope of this thesis, such as reputation or trust may suffice as a proxy for or even improve the perceived quality of the data.[5]

That some needs may be met by attributing to an intermediary source, however, does not preempt the need to know more. We might still wish to look beyond the integrator, unfolding layers of reuse and redistribution back to the underlying initial data sources. We therefore propose an algorithm for unfolding an attribution by recursively attributing values in the intermediary. Based upon this algorithm, we conclude that we can compose an attribution in the same way that we compose relational queries.

Example 3.9 Attribution composition

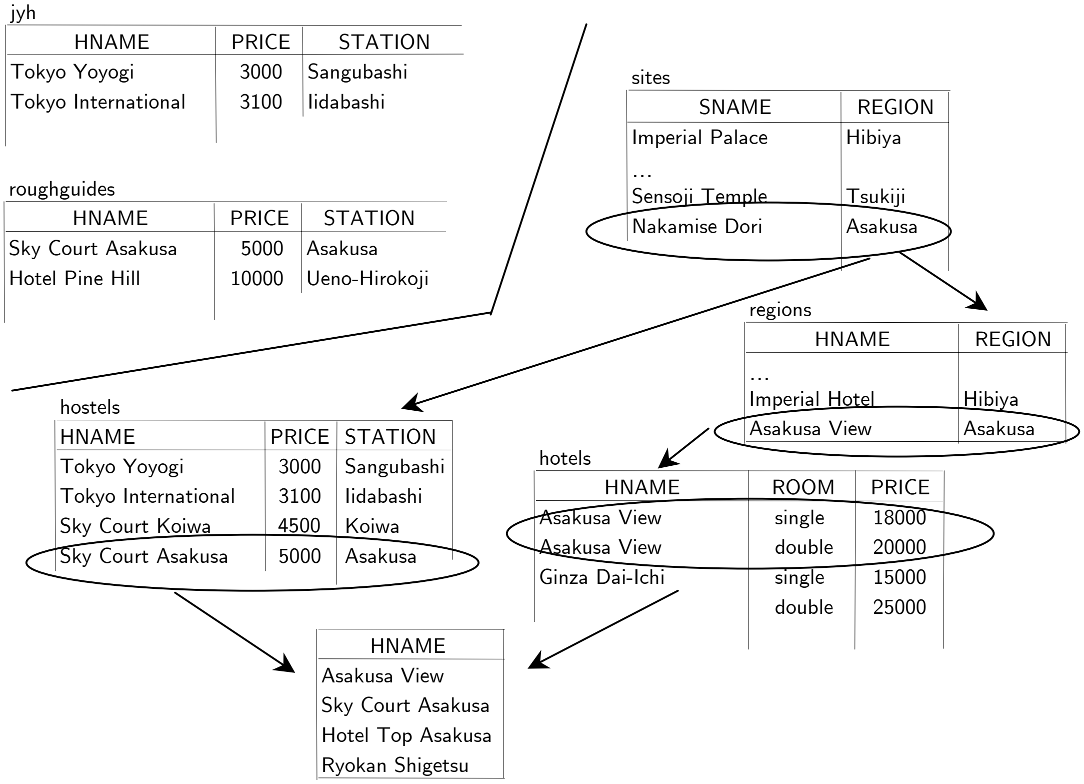
Refer again to Q5 from example 3.8. We can translate the SQL queries into:

DRC5.1 {HNAME | (hotels(HNAME, ROOMS, PRICE) ∧ regions(HNAME, REGION) ∧ sites("Nakamise Dorsi", REGION)) ∨ (hostels(HNAME, PRICE, STATION) ∧ sites("Nakamise Dorsi", STATION))}

DRC5.2 {HNAME | (hotels(HNAME, ROOMS, PRICE) ∧ regions(HNAME, REGION) ∧ sites("Nakamise Dorsi", REGION)) ∨ (jyh(HNAME, PRICE, STATION, PHONE, FAX) ∧ sites("Nakamise Dorsi",STATION)) ∨ (rg(HNAME, PRICE, STATION, PHONE) ∧ sites("Nakamise Dorsi",STATION))}

Recall also that predicate hostels(XYZ) in DRC5.1 corresponds to hostels(HNAME, PRICE, STATION, PHONE) ≝ jyh(HNAME, PRICE, STATION, PHONE, FAX) ∨ rg(HNAME, PRICE, STATION, PHONE)

[5] We hypothesize that the data source provides a heuristic for the quality (e.g. timeliness or veracity) of data available from the source. Data that comes from an unknown database producer may benefit (or suffer) from integration and redistribution by compounding the positive (or negative) reputation of the integrator. If an unknown data source is cited in the Wall Street Journal, the perceived quality of the data might rise whereas if the data is cited in a daily tabloid known for exaggeration or hyperbole, the perceived quality of the data might fall.



jyh

HNAME	PRICE	STATION
Tokyo Yoyogi	3000	Sangubashi
Tokyo International	3100	Iidabashi

roughguides

HNAME	PRICE	STATION
Sky Court Asakusa	5000	Asakusa
Hotel Pine Hill	10000	Ueno-Hirokoji

sites

SNAME	REGION
Imperial Palace	Hibiya
…	
Sensoji Temple	Tsukiji
Nakamise Dori	Asakusa

regions

HNAME	REGION
…	
Imperial Hotel	Hibiya
Asakusa View	Asakusa

hostels

HNAME	PRICE	STATION
Tokyo Yoyogi	3000	Sangubashi
Tokyo International	3100	Iidabashi
Sky Court Koiwa	4500	Koiwa
Sky Court Asakusa	5000	Asakusa

hotels

HNAME	ROOM	PRICE
Asakusa View	single	18000
Asakusa View	double	20000
Ginza Dai-Ichi	single	15000
	double	25000

HNAME
Asakusa View
Sky Court Asakusa
Hotel Top Asakusa
Ryokan Shigetsu

Figure 3.4 Query composition

Regardless of how the query is posed, the result is the list of hotels and hostels: Asakusa View, Ryokan Shigetsu, Sky Court Asakusa, and Hotel Top Asakusa

Step 1 in the algorithm is to collect the substitutions for the composed query, DRC 5.1. For brevity, we will only illustrate the composition of the relevant substitution. The relevant substitutions are:

{<hotels("Asakusa View"/HNAME); regions("Asakusa View"/HNAME,
    "Asakusa"/REGION); sites("Nakamise Dorsi"/SNAME, "Asakusa"/REGION)>;
<hostels("Ryokan Shigetsu"/HNAME, "Asakusa"/STATION); sites("Nakamise
    Dorsi"/SNAME, "Asakusa"/STATION)>;
<hostels("Sky Court Asakusa"/HNAME; "Asakusa"/STATION); sites("Nakamise
    Dorsi"/SNAME, "Asakusa"/STATION)>;
<hostels("Hotel Top Asakusa"/HNAME) "Asakusa"/STATION); sites("Nakamise
    Dorsi"/SNAME, "Asakusa"/STATION)>}

Informally, in Step 2 of the algorithm, we find the variables applicable to the composed relation, hostels and attribute those values against DRC 4.2. Yielding the following substitutions:
{<rg("Ryokan Shigetsu"/HNAME)>;

<rg("Sky Court Asakusa"/HNAME)>;
<rg("Hotel Top Asakusa"/HNAME)>;
<jyh("Sky Court Asakusa"/HNAME)>}

To complete the attribution composition, in Step 3, we combine the respective substitutions:

{<hotels("Asakusa View"/HNAME); regions("Asakusa View"/HNAME,
    "Asakusa"/REGION); sites("Nakamise Dorsi"/SNAME, "Asakusa"/REGION)>;
<rg("Ryokan Shigetsu"/HNAME, "Asakusa"/STATION); sites("Nakamise Dorsi"/SNAME,
    "Asakusa"/STATION)>;
<rg("Sky Court Asakusa"/HNAME; "Asakusa"/STATION); sites("Nakamise
    Dorsi"/SNAME, "Asakusa"/STATION)>;
<rg("Hotel Top Asakusa"/HNAME) "Asakusa"/STATION); sites("Nakamise
    Dorsi"/SNAME, "Asakusa"/STATION)>;
<jyh("Sky Court Asakusa"/HNAME; "Asakusa"/STATION); sites("Nakamise
    Dorsi"/SNAME, "Asakusa"/STATION)>}

This ultimately translates to the following relevant attribution:

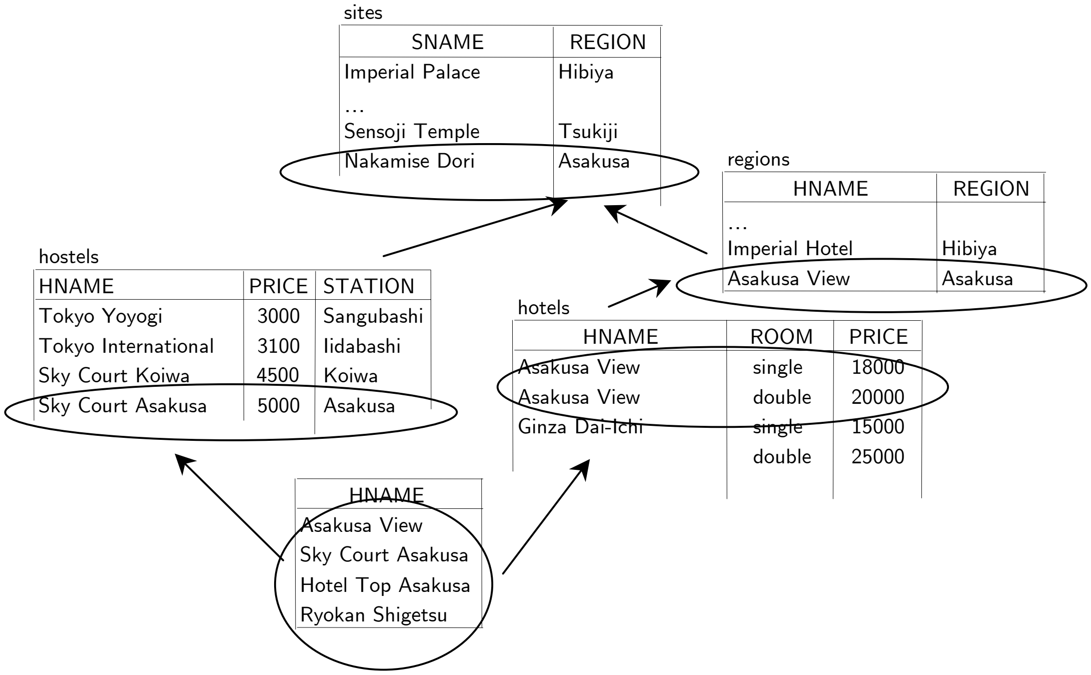
{<hotels; regions; sites>; <rg; sites>; <jyh; sites>}

The process of composing an attribution by iteratively tracing backwards through the constituent inputs is depicted in Figure 3.5.□

In looking more closely at Examples 3.6 and 3.9, we see that certain data values, such as the hostel "Sky Court Asakusa" may appear multiple times. This observation hints at a second category of multiple derivations, those within a single expression.

We originally separated multiple derivations into two categories: derivations from multiple expressions and derivations within a single expression. We can further separate derivations from a single expression into cases of weak equivalence and cases of natural join.

Weak equivalence encompasses the idea that tuples in a query result may differ only in their attribution. A straightforward example of this occurs in the case of relational union.



sites

SNAME	REGION
Imperial Palace	Hibiya
…	
Sensoji Temple	Tsukiji
Nakamise Dori	Asakusa

regions

HNAME	REGION
…	
Imperial Hotel	Hibiya
Asakusa View	Asakusa

hostels

HNAME	PRICE	STATION
Tokyo Yoyogi	3000	Sangubashi
Tokyo International	3100	Iidabashi
Sky Court Koiwa	4500	Koiwa
Sky Court Asakusa	5000	Asakusa

hotels

HNAME	ROOM	PRICE
Asakusa View	single	18000
Asakusa View	double	20000
Ginza Dai-Ichi	single	15000
	double	25000

HNAME
Asakusa View
Sky Court Asakusa
Hotel Top Asakusa
Ryokan Shigetsu

Figure 3.5 Attribution composition: Step 1

Example 3.10 Weak equivalence in union

SQL 4.1 select * from hostels

SQL 4.2 select HNAME, PRICE, STATION from jyh
union
select HNAME, PRICE, STATION from rg

in SQL 4.1, there is one substitution associated with Sky Court Asakusa
g("Sky Court Asakusa"/HNAME, 5000/PRICE, "Asakusa"/STATION, "81-3-3672-4411"/PHONE)

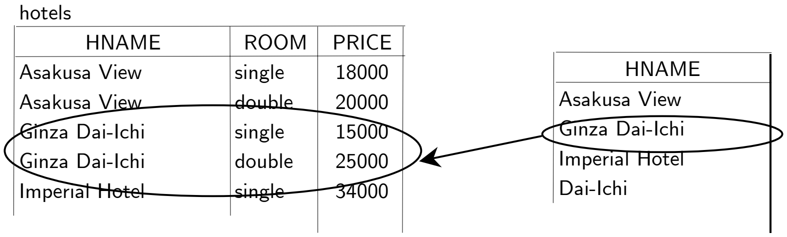
But in 4.2 there are TWO, one associated w/ querying rg (*r*) and one associated w/ querying jyh (*s*)
r("Sky Court Asakusa"/HNAME, 5000/PRICE, "Asakusa"/STATION, "81-3-3672-4411"/PHONE);
s("Sky Court Asakusa"/HNAME, 5000/PRICE, "Asakusa"/STATION, "81-3-3672-4411"/PHONE, "81-3-3875-4941"/FAX)} □

Similar behavior is exhibited when projecting a list of attributes that do not constitute a candidate key.

Example 3.11 Weak equivalence in projection

SQL 3.2 select HNAME from hotels

Pick one of the hotels in the result, for example. As seen in Figure 3.6, for each HNAME in the relation hotels, There are two lists of substitutions:
{("Ginza Dai-Ichi"/HNAME, "single"/ROOMS, 15000/PRICE)>;
<("Ginza Dai-Ichi"/HNAME, "double"/ROOMS, 25000/PRICE)>} □



hotels

HNAME	ROOM	PRICE
Asakusa View	single	18000
Asakusa View	double	20000
Ginza Dai-Ichi	single	15000
Ginza Dai-Ichi	double	25000
Imperial Hotel	single	34000

HNAME
Asakusa View
Ginza Dai-Ichi
Imperial Hotel
Dai-Ichi

Figure 3.6 Weak duplicates

Though logical models of relations, like the relational calculus, rely upon set semantics, this theory of attribution treats every instance of a tuple as unique and having a distinct attribution with respect to the query and underlying data sources. To preserve the set semantics of the relational data model, attributions for weak duplicates are combined together.

The second category of duplication, that occurs in single expressions, stems from looking for relationships between relations (called a join operation) rather than taking the union of different relations. Informally, we want to distinguish between comparisons on values that represent the same thing and values that merely "look" alike.[6] We call values that represent the same thing duplicates. However, we would like to treat values that merely "look" alike somewhat differently.

Example 3.12 Multiple derivations in joins.

To explore this issue, we will reconsider Q5 from earlier. However, this time, we separate the query explicitly into:
Q6. Identify hotels around the Nakamise Dori and
Q7. Identify hostels around the Nakamise Dori.
These queries translate to SQL 6 and SQL 7, as indicated below. Separating the queries this way will allow us to look more carefully at how values are compared between tables.

[6] We consider natural join as distinct from theta comparison where theta is equality. Natural join is represented in the relational calculus as multiple occurrences of the same variable in two or more predicates. In the (named) relational algebra, it corresponds to the idea that different relations may include the same domain. Using a slight variation on the standard notation, this is represented by identical attribute names in multiple relation schemes.

```
SQL6    select HNAME
        from hotels, regions, sites
        where hotels.HNAME = regions.HNAME
        and regions.REGION = sites.REGION
        and sites.SNAME = "Nakamise Dorsi"

SQL7    select HNAME
        from hostels, sites
        where hostels.STATION = sites.REGION
        and sites.SNAME = "Nakamise Dorsi"
```

We translate the above SQL queries into the following DRC expressions:

DRC 6 {HNAME | hotels(HNAME, ROOMS, PRICE) ∧ regions(HNAME, REGION) ∧ sites("Nakamise Dorsi", REGION)}

DRC 7.1 {HNAME | hostels(HNAME, PRICE, STATION) ∧ sites("Nakamise Dorsi", STATION)}

DRC 7.2 {HNAME | hostels(HNAME, PRICE, STATION) ∧ sites("Nakamise Dorsi", REGION) ∧ (STATION = REGION)}

(where DRC 6 and DRC 7.1 are the subformulas that we used in DRC 5 and DRC 7.2 is a logically equivalent expression to DRC 7.1

To find hotels around Nakamise Dorsi, we use geographic region names associated with tourist attractions and also associated with the hotel addresses. Unfortunately, we do not have such information available for the youth hostels. Instead, we match the regions for the local tourist attractions with the names of railroad stations. This is illustrated in Figure 3.7. The scalar values are the same, but they come from different domains. This distinction is made explicit in the calculus by the distinction between multiple occurrences of the same variable versus explicit equality. Consider a few of the comprehensive substitutions for the expressions from Example 3.12.

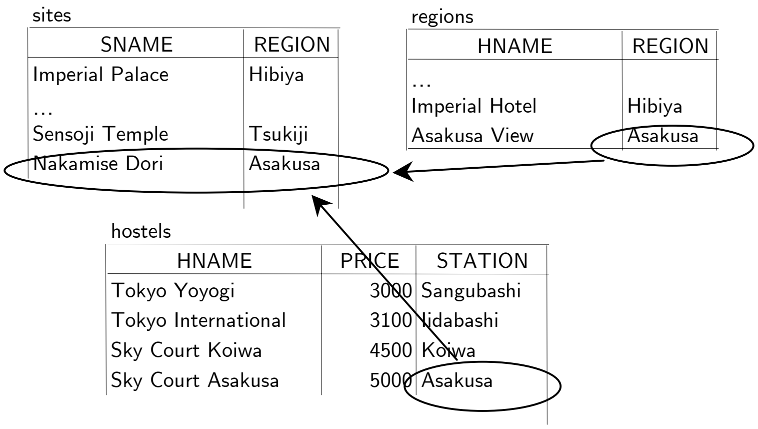
comprehensive substitution for DRC 6
<"Asakusa View"/HNAME, "single"/ROOMS, 18000/PRICE, "Nakamise Dorsi"/SNAME, "Asakusa"/REGION>

comprehensive substitution for DRC 7.1
<"Ryokan Shigetsu"/HNAME, 7000/PRICE, "Asakusa"/STATION, "Nakamise Dorsi"/SNAME>

comprehensive substitutions for DRC 7.2
<"Ryokan Shigetsu"/HNAME, 7000/PRICE, "Asakusa"/STATION, "Nakamise Dorsi"/SNAME, "Asakusa"/REGION>

The intuition is that multiple occurrences of the same variable constitute a *single substitution* that derives from multiple sources. The substitution "Asakusa"/REGION in

DRC 6 stems from two distinct sources; hotels and region. Explicit equality, by contrast, suggests that the equated variables are different values with their own substitution. The substitution "Asakusa"/REGION is equated with the substitution "Asakusa"/STATION in DRC 7, but we do not consider these substitutions to share the same sources. The relation hostels is not a source for "Asakusa"/REGION even though the variables are equated and the relation hotels is considered a source.[7] □



sites

SNAME	REGION
Imperial Palace	Hibiya
...	
Sensoji Temple	Tsukiji
Nakamise Dori	Asakusa

regions

HNAME	REGION
...	
Imperial Hotel	Hibiya
Asakusa View	Asakusa

hostels

HNAME	PRICE	STATION
Tokyo Yoyogi	3000	Sangubashi
Tokyo International	3100	Iidabashi
Sky Court Koiwa	4500	Koiwa
Sky Court Asakusa	5000	Asakusa

Figure 3.7 Multiple derivations in joins

Note the implicit equivalence between multiple occurrences of the same variable versus the explicit built-in theta-comparison predicate (X=Y) in calculus expressions. We arrive at this conclusion by substituting all occurrences of Y with X and eliminating the explicit theta-comparison. Because our intuition for attribution makes use of the distinction despite the implicit equivalence, we conclude that the different types of attribution are not equivalent for equivalent expressions when we allow built-in predicates for explicit equality.

Negation is the other place in which our observations on the attribution of equivalent queries breaks down. Our intuition is that attribution corresponds to those substitutions that correspond to a true interpretation. What then is the substitution that makes a statement about the non-existence of something true? Applying the conventional database interpretation of negation, we suggest that the way to prove a negative substitution is by comparing that substitution to all known positive substitutions. If the item of interest is not known to be true, we conclude that it must be false.[8]

[7] Note that railroad station names and geographic region names may not always coincide. The example here is intended to illustrate situations where values from different domains are used in comparisons and query conditions, suggesting distinct lineage.

[8] The negation as failure interpretation adopted in the database community suggests that a negated subformula is true only when no true interpretation of the subformula is found (Ullman 1988).

Example 3.13 Negation

We take our original query and invert it.

Q8. Hotels NOT by the Imperial Palace. We can write this in SQL as:

```
SQL 8.1   select HNAME
          from regions
          where HNAME not in (
                    select HNAME
                    from regions, sites
                    where regions.REGION = sites.REGION
                    and sites.SNAME = "Imperial Palace")
```

One possible interpretation of this expression in the DRC is:

DRC 8.1 {HNAME | regions(HNAME, REGION) ∧ ¬(regions(HNAME, REGION) ∧
sites("Imperial Palace", REGION)}

The answer to DRC 8.1 is: Hotel Sofitel, Katsutaro, Asakusa View

We know that the Asakusa View is not by the Imperial Palace. What are the corresponding substitutions into DRC8.1 indicating the truth of this? We need to establish, in a positive sense, what hotels are by the Imperial Palace and then, given a fixed list of hotels, (those in regions), we keep the remainder. This process is depicted in Figure 3.8.

Consider, for brevity, just one comprehensive attribution for DRC 8.1:
<*f*("Asakusa View"/HNAME, "Asakusa"/REGION)
g("Dai-Ichi"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME)
g("Imperial"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME)> □

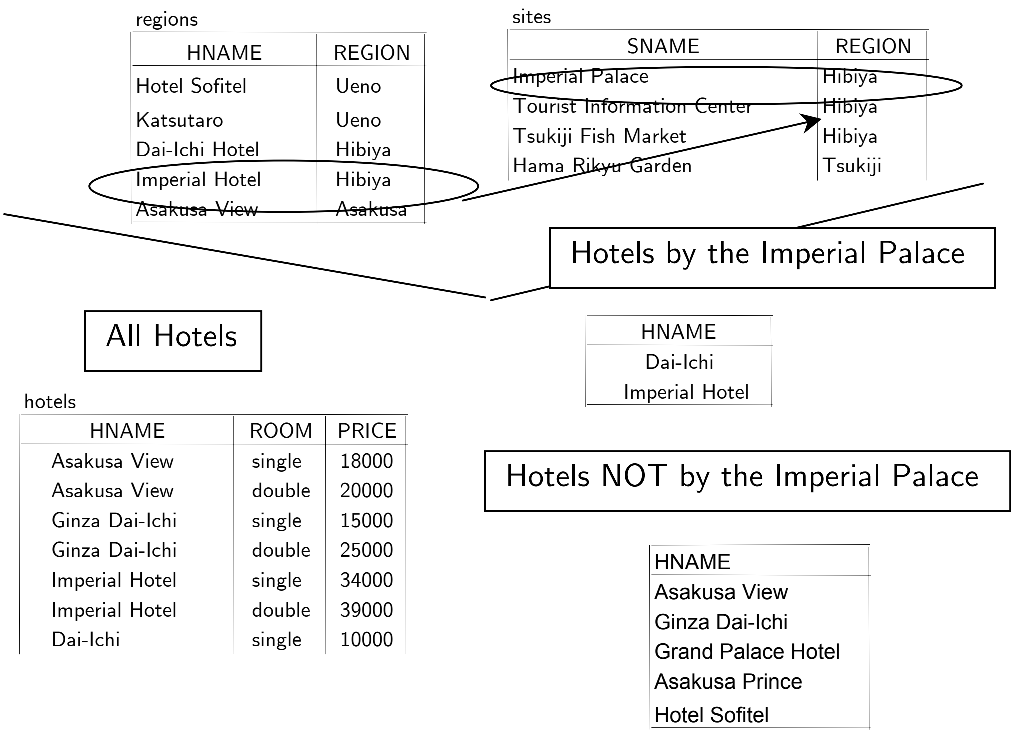
To see some of the difficulty created by negation, consider an equivalent expression to DRC 8.1, which we present in Example 3.14.

Example 3.14 Attribution equivalence breaks down under negation

DRC 8.2 {HNAME | regions(HNAME, REGION) ∧ ¬sites("Imperial Palace", REGION)}

DRC 8.2 is logically equivalent to DRC 8.1. The difference is that we have pushed the negation down to the atoms and then distributed the conjuncts and disjuncts.

Compare the comprehensive substitution associated with the hotel Asakusa View that we examined in Example 3.13. Notice
<*f*("Asakusa View"/HNAME, "Asakusa"/REGION)
g("Imperial Palace"/SNAME, "Hibiya"/REGION)> □



regions

HNAME	REGION
Hotel Sofitel	Ueno
Katsutaro	Ueno
Dai-Ichi Hotel	Hibiya
Imperial Hotel	Hibiya
Asakusa View	Asakusa

sites

SNAME	REGION
Imperial Palace	Hibiya
Tourist Information Center	Hibiya
Tsukiji Fish Market	Hibiya
Hama Rikyu Garden	Tsukiji

Hotels by the Imperial Palace

HNAME
Dai-Ichi
Imperial Hotel

All Hotels

hotels

HNAME	ROOM	PRICE
Asakusa View	single	18000
Asakusa View	double	20000
Ginza Dai-Ichi	single	15000
Ginza Dai-Ichi	double	25000
Imperial Hotel	single	34000
Imperial Hotel	double	39000
Dai-Ichi	single	10000

Hotels NOT by the Imperial Palace

HNAME
Asakusa View
Ginza Dai-Ichi
Grand Palace Hotel
Asakusa Prince
Hotel Sofitel

Figure 3.8 Negation in attribution

3.3 Levels of attribution

In our last two Examples 3.13 and 3.14 we examined only a subset of the substitutions. In particular, we considered only those substitutions corresponding to the result that the hotel Asakusa View is not by the Imperial Palace. This suggests that, rather than speaking about the attribution for a query result, we might wish to consider attributing only one part of the result. Returning to the analogy of a bibliography, perhaps the reader is only interested in Part 1 of the thesis. As noted in the Introduction, Part 1 consists of Sections 3, 4, 5, and 6. Taken together, these four Sections present our model of attribution. However, our bibliography, which follows Section 9, is a single list of all works referenced throughout the entire document. The reader might only wish to know the works which were referenced in Section 3 – 6. Perhaps the reader is only interested in Section 3.

And because we know that the result of one query can become the source for another query (query composition), we extend the idea of attributing one part of the result to the idea that we might attribute with only part of a source rather than attribute using the relation as a whole. We refer to these ideas as result and source granularity respectively.

The general intuition is that attribution defines pointers or references from a query result back to its constituent sources. Granularity therefore corresponds to the pointer's precision. Beginning with result granularity, at the finest granularity, we might wish to attribute a specific instance of a value in a result. More generally, we might think of all instances of a value in a result. Further coarsening our granularity, we could consider the attribution corresponding to a range of values (e.g. an entire tuple, a set of tuples, or perhaps a column). At the limit, we could attribute the entire result relation.

Example 3.15 Result granularity

Consider Q9, Hotel names, hotel prices, and names of sites around Tokyo, Japan.

SQL 9 select HNAME, PRICE, SNAME
from hotels, sites

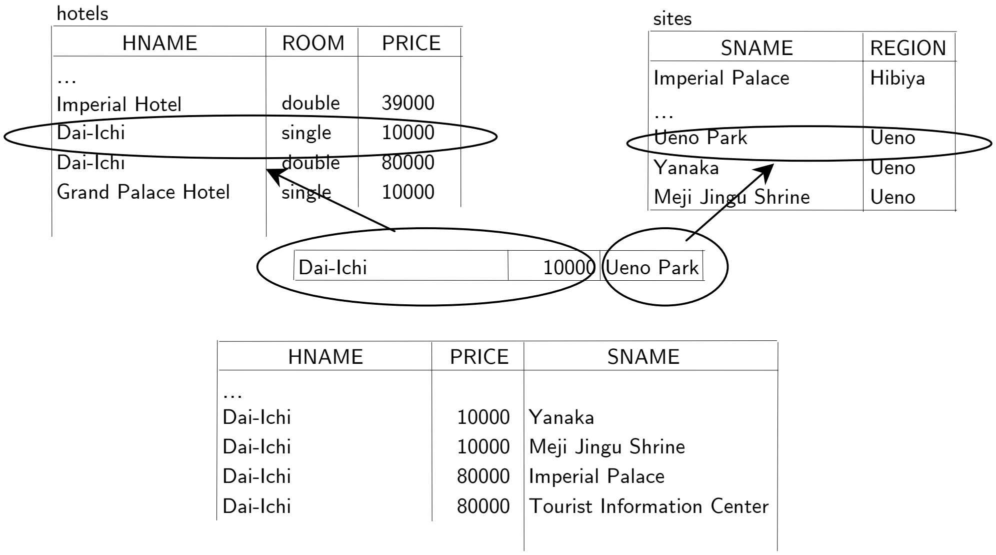
DRC 9.1 {HNAME, PRICE, SNAME | hotels(HNAME, ROOM, PRICE) $\wedge$ sites(SNAME, REGION)}

As seen in Figure 3.9, we can discuss the attribution associated with the specific instance of a result where HNAME = "Dai-Ichi" (corresponding to one tuple). Single rooms by Ueno Park correspond to the following tuple: <"Dai-Ichi", 10000, "Ueno Park">

The corresponding comprehensive attribution is:
{<*f*("Dai-Ichi"/HNAME, "single"/ROOM, 10000/PRICE, "Ueno Park"/SNAME, "Ueno"/REGION)>}

We could ask for all instances of "Dai-Ichi" hotel in the result. Because the query is a Cartesian product, the actual solution is quite large, but one part of it includes Table 3.6 with the following substitutions:
{… < *f*("Dai-Ichi"/HNAME, "single"/ROOM, 10000/PRICE, "Yanaka"/SNAME, "Ueno"/REGION)>;
f("Dai-Ichi"/HNAME, "single"/ROOM, 10000/PRICE, "Meji Jingu Shrine"/SNAME, "Ueno"/REGION)>;
f("Dai-Ichi"/HNAME, "double"/ROOM, 80000/PRICE, "Imperial Palace"/SNAME, "Hibiya"/REGION)>;
f("Dai-Ichi"/HNAME, "double"/ROOM, 80000/PRICE, "Tourist Information Center"/SNAME, "Hibiya"/REGION)>; …} □

Figure 3.9 Source/result granularity



hotels

HNAME	ROOM	PRICE
…		
Imperial Hotel	double	39000
Dai-Ichi	single	10000
Dai-Ichi	double	80000
Grand Palace Hotel	single	10000

sites

SNAME	REGION
Imperial Palace	Hibiya
…	
Ueno Park	Ueno
Yanaka	Ueno
Meji Jingu Shrine	Ueno

Dai-Ichi	10000	Ueno Park

HNAME	PRICE	SNAME
…		
Dai-Ichi	10000	Yanaka
Dai-Ichi	10000	Meji Jingu Shrine
Dai-Ichi	80000	Imperial Palace
Dai-Ichi	80000	Tourist Information Center

Dai-Ichi	10000	Yanaka
Dai-Ichi	10000	Meji Jingu Shrine
Dai-Ichi	80000	Imperial Palace
Dai-Ichi	80000	Tourist Information Center

Table 3.6 Result granularity

Likewise, we might draw the parallel conclusions for source granularity. We could attribute using a specific instance of a substitution (e.g. the source tuple that corresponds to the specific instance of a substitution), all occurrences of a substitution in a particular source (e.g. every tuple in a source that provides the substitution), or again at the extreme, the name of the relation that corresponds to true substitutions.

Example 3.16 Source granularity

Throughout the Section, we have given answers for sources as relation names. Using DRC 9 from Example 3.15, however, we can provide references to the sources with varying levels of precision as well.

Attribution for DRC 9 as sources:
<hotels; sites>

We can also give:
hotels("Dai-Ichi"/HNAME)

which implicitly indicates all instances in the hotels relation where HNAME = "Dai-Ichi"
{<"Dai-Ichi", "single", 10000>; <"Dai-Ichi", "double", 80000>}

or we can give an explicit instance of "Dai-Ichi" in the source relation
<hotels("Dai-Ichi"/HNAME, "single"/ROOM, 10000/PRICE)> □

Given that we expressed our intuition about attribution in terms of an expression and a result, how might we express interest in the attribution for an explicit granule rather than the relation as a whole, given that we have been thinking about attribution in terms of answers to queries? Conceptually, we know that we can think of substitutions that make a particular substitution for the free variables (one tuple in the result) true. However, within our framework of attribution for relations, we might also take our cue from the observation that the relational calculus is closed. Closure permits us, as demonstrated earlier, to compose queries. At the same time, we know that we can compose attribution as well. Consequently, if we want all instances or specific instances of values in the result, we propose composing a query on the result and then composing the corresponding attribution to return the attribution for the result granule of interest.

Example 3.17 Specifying granularity

We refer again to Q9 Hotel names, hotel prices, and names of sites around Tokyo, Japan.

SQL 9 select HNAME, PRICE, SNAME
from hotels, sites

Intuitively, if we are interested in a result granule defined as, all instances of "Dai-Ichi" in the result, we think of something like:

DRC 9.2 {"Dai-Ichi", PRICE, SNAME | hotels(HNAME, ROOM, PRICE) $\wedge$ sites(SNAME, REGION)}

In other words, we want all substitutions in the answer where the HNAME is "Dai-Ichi." We can construct just such a query if we think of:

DRC 10 {HNAME, ROOM, PRICE, SNAME | temp("Dai-Ichi", ROOM, PRICE, SNAME) $\wedge$ (HNAME = "Dai-Ichi")}

Where temp(HNAME, ROOM, PRICE, SNAME) $\stackrel{\text{def}}{=}$ {HNAME, ROOM, PRICE, SNAME | hotels(HNAME, ROOM, PRICE) $\wedge$ sites(SNAME, REGION)}

We might also think of a subset of instances of "Dai-Ichi" in the result. Consider:

DRC 11 {HNAME, ROOM, PRICE, SNAME | temp(HNAME, ROOM, PRICE, SNAME) $\wedge$ (HNAME = "Dai-Ichi") $\wedge$ (ROOM = "single") $\wedge$ (PRICE = 10000)} □

Regardless of source granularity, the comprehensive attribution for a value in result tuple is the same for every other value in the same tuple. This makes sense. A DRC expression corresponds to a set of tuples. Therefore, one list of substitutions that makes one instance of the expression true applies to every value in the corresponding result tuple. Likewise, given relation-level source granularity, the comprehensive attribution for every value in the result is the same. Again this makes intuitive sense. This merely articulates the observation that all of the relations in the WHERE clause of an SQL statement apply to the relation as a whole.

Note by contrast that for source or relevant attribution, the attribution of different values or tuples are not necessarily the same. In the UNION case, we saw how weak duplicates illustrated a single tuple might have more than one source. As a more subtle case, refer again to the Cartesian product of Q9. From Figure 3.9, we see how distinct sources can associate with only a subset of attributes in a result relation.

In summary, we list the different features and properties captured by our model of attribution and discussed throughout this Section. We present the model more formally in Section 4 and subsequently propose an extension to the relational algebra to operationalize one instantiation of our theory. In particular, we demonstrate attribution using relation-level source granularity. We conclude Part 1 by considering how the theory might extend into the semi-structured environment of the Web.

- Attribution refers to the substitutions that make the interpretation of the expression true.
- In the case of negation, we use negation as failure semantics to establish that a predicate does not hold.
- There are three distinct types of attribution: comprehensive, source, and relevant.
- There are a number of ways in which a query result might have more than one attribution:
    - Multiple queries for the same result
    - Weak equivalence
    - Strict equivalence (equivalent expressions using only base relations)
    - Equivalence using composed data sources
    - Weak duplicates
    - Multiple instances of the same variable in an expression (e.g. natural join)
- For conjunctive queries with theta-comparisons but omitting explicit equality, the comprehensive and source attribution of equivalent queries is equivalent.
- For positive queries, the relevant attribution of equivalent queries is equivalent
- We can compose the attribution of composed queries (where there is no more than one level of negation) by recursively unfolding and attributing sub-queries in a depth-first manner.
- Weak duplicates and multiple occurrences of the same value in different predicates of a calculus expression (join variables in a natural join) entail multiple derivations of the same row or column (tuple or attribute domain).
- Theta comparisons involving explicit equality represent different values, each with their own, distinct derivation.
- We can attribute using different levels of granularity on the source side and attribute different result granules.

- We specify the attribution of different result granules by composing queries on the original result.

4 Formal model

In this Section, we present our formal model of attribution along with a number of properties of the model. Section 1 offers a brief overview of the domain relational calculus for those unfamiliar with the formalism. Section 2 introduces our definition of attribution in the context of the syntax of a domain relational calculus expression. To formalize the model, we begin with the set of conjunctive queries (defined below) and gradually expand the query language in the standard way.[9] We conclude the Section by relating the formal model back to the desiderata originally specified in Section Two.

4.1 The domain relational calculus

Our formalization of attribution is based upon the Domain Relational Calculus (DRC). For those already familiar with the DRC, we begin by specifying the calculus syntax and notation used in the remainder of this thesis. For those unfamiliar with the DRC, we follow our specification with a brief overview. The DRC is built upon, and our overview assumes, basic familiarity with the first-order predicate calculus.

4.1.1 Syntax and notation

We use the set of lists notation for a relation. Following (Ullman 1988; 1989), at times, we make selective use of variable names to denote attributes for readability. We define attribution in terms of the interpretation (Maier 1983) of a safe DRC expression, where safety is defined syntactically by (Ullman 1988).

A list of substitutions $a = <c_1/X_1, c_2/X_2, \ldots, c_n/X_n>$ projected on a formula f, written $a(f)$, returns the sub-list of substitutions for the variables in f. A list of substitutions a is in the attribution for an expression $E = \{x \mid f(x)\}$ when a has the minimal number of substitutions required to recursively interpret every sub-formula $f'$of f such that $s(f) = c$ and $I(f\,(c/x)) = true$.

Furthermore, all expressions are assumed to be in Safe Range Normal Form (SRNF) and Relational Algebra Normal Form (RANF) meaning negations are pushed down to atoms and existential quantification and connectives are flattened (Abiteboul, Hull, and Vianu 1995). We further assume, consistent with RANF, that formulas without negation are expanded into prenex disjunctive normal form (DNF). Given the syntactic safety rules, each disjunct therefore projects all and only the set of free variables in the expression.

As a shorthand, for expressions of the form:
$\{X_1,X_2,\ldots,X_n \mid \exists Y_1, \exists Y_2,\ldots,\exists Y_m)\;\; f(X_1,X_2,\ldots,X_n,Y_1,Y_2,\ldots,Y_m)\}$
We will sometimes substitute:
$\{X_1,X_2,\ldots,X_n \mid (\exists Y_1,Y_2,\ldots,Y_m)\;\; f(X_1,X_2,\ldots,X_n,Y_1,Y_2,\ldots,Y_m)\}$
And when obvious, we will omit the existential quantification entirely:
$\{X_1,X_2,\ldots,X_n \mid f(X_1,X_2,\ldots,X_n,Y_1,Y_2,\ldots,Y_m)\}$

[9] See (Ullman 1988) and (Abiteboul, Hull, and Vianu 1995).

Following (Ullman 1988), we use Extensional Data Base (EDB) to refer to base relations and Intentional Data Base (IDB) to refer to relations composed on base relations (e.g. views).

4.1.2 A review of the calculus

Let **D** be a set of disjoint domains over which all relations are defined. A relational scheme is a pair (J,D) where J is an index (a set of integers from 1 to $max(J)$) and D is a function, $D: J \rightarrow \mathbf{D}$. A relation is then defined over a scheme as a finite subset of the Cartesian Product of the domains in the scheme. A tuple is therefore a list of values where the J^{th} value is drawn from the corresponding domain and a relation is a finite set of such lists.

In practice, the set-of-lists notation is equivalent to more conventional attribute-value naming (Ullman 1988). Following Ullman and (Abiteboul, Hull, and Vianu 1995), where obvious to do so, we may use carefully selected variable names to denote particular attribute domains. We may then denote a relation scheme by a tuple instance consisting entirely of domain variables, an ordered list of variable names $(A_1,A_2, \ldots ,A_{max(J)})$ where each A_J is a variable name for a value drawn from $D(J)$.

Harkening back to our motivating example from Section 3, variable names might include: NAME, PRICE, REGION, ROOM, STATION, etc.

Definition 4.1 Atomic formulas
Basic formulas in the domain calculus (also called *atomic formulas*) are expressed in terms of relations, domain variables, and Θ, the set of comparison operators (e.g. >, ≥ , =, ≤, <) for every domain in **D**.

1. If r is a relation in d with scheme $(A_1,A_2, \ldots ,A_n)$ then $r(X_1,X_2,\ldots,X_n)$ is an atomic formula where X_i is either a domain variable for D_i (e.g. of type D_i) or a constant $c_i \in D_i$.
2. If X and Y are domain variables and c is a constant drawn from the appropriate domain, then $X\ \theta\ Y$, $X\ \theta\ c$, and $c\ \theta\ X$ are all atomic formulae.
3. The Boolean constants *true* and *false* are also atomic formulae. □

Example 4.1 Atomic formulas
hotels(HNAME,ROOM,PRICE) is a predicate for the relation hotels
hotels('Asakusa View', 'single', 18000) is an atomic formula
hotels (HNAME, 'single', PRICE) is also an atomic formula as are
hotels.HNAME = regions.HNAME and PRICE < 90,000. □

Definition 4.2 Calculus formula
We recursively extend our definition of a calculus formula by building upon our atomic formulae using the logical connectives (¬, ∧, ∨) and the quantifiers (∃, ∀) in a manner similar to the predicate calculus.

1. If f is a formula, then $\neg f$ is a formula.
2. If f and g are both formulas, then $f \wedge g$ is a formula as is $f \vee g$.
3. If f is a formula and X is a domain variable, then $\exists X f$ and $\forall X f$ are both formulas where free occurrences of X in f are bound by $\exists X$ and $\forall X$ respectively using the expected definitions for free and bound (Maier 1983 at 231; Ullman 1988 at 147).
4. If f is a formula, then (f) is a formula

The parentheses explicitly define groupings of operands as we might expect. In the absence of parentheses, the quantifiers $\exists X$, and $\forall X$ have highest, equal precedence. $\neg$, $\wedge$, $\vee$ follow in decreasing order of precedence. □

Example 4.2 Calculus formulas
If regions(HNAME, Hibya) is a formula, then ¬ regions(HNAME, Hibya) is a formula.
It therefore follows that (¬ regions(HNAME, 'Hibya')) is a formula.
(¬ regions(HNAME, 'Hibya')) ∧ hotels(HNAME,ROOM,PRICE) is also a formula.
Using the quantifiers in conjunction with parentheses can result in some subtly different formulas.
∃HNAME(¬ regions(HNAME, 'Hibya')) ∧ hotels(HNAME,ROOM,PRICE) is not equal to
∃HNAME((¬ regions(HNAME, 'Hibya')) ∧ hotels(HNAME,ROOM,PRICE)). □

We offer a brief aside on the legality of formulas and note that domain variables should be used consistently so that in the formula ∃X((¬ regions(X, 'Hibya')) ∧ hotels(X,Y,Z)), domain variable X refers to the domain of lodging establishment names and the formula ∃X((¬ regions(X, 'Hibya')) ∧ hotels(Z, X, X)) is somewhat nonsensical (Maier 1983 at 231).

Given a formula f, we would like to know what that formula means. Following Maier, we first define a *substitution*. We then arrive at an *interpretation* of f based upon a substitution for the free variables in f and the expected meaning of the logical connectives and quantifiers. The following definitions for substitutions and interpretations are the foundation of our formalism for attribution.

The intuition behind the substitution is to recall that formulas are defined with respect to a set of base relations called a database d. Atomic formulas for relation r in database d correspond to base tables in d (or constraints that take the form of comparisons on values that appear in one or more initial tables.) A substitution is a "random" replacement of all free variables in a formula with constants from their corresponding attribute domains. An atomic formula denoted by $r(X_1,X_2,\ldots,X_n)$ is *true* for all and only the substitutions $(c_1,c_2,\ldots,c_n)$ that are in the base table $r \in d$.

Definition 4.3 Substitution
More formally, let $f(X_1,X_2,\ldots,X_n)$ be a legal calculus formula as defined earlier where $X_1,X_2,\ldots,X_n$ corresponding to their respective domains are the only free domain variables in f.

A *substitution* of a tuple $(c_1,c_2,...,c_n)$ in $f(X_1,X_2,...,X_n)$ is denoted by $f(c_1/X_1,c_2/X_2,..., c_n/X_n)$ where $c_i \in D_i$, the domain corresponding to A_i. We rewrite *f*, replacing every free occurrence of X_i with c_i. Ground atoms, atomic formulae containing only constants ($\underline{k_i}$) following the substitution, are replaced with *true* or *false* as follows:

1. If the ground atom is a relation $r(k_1,k_2,...,k_m)$ then replace the atom in the formula *f* with *true* if tuple $(k_1,k_2,...,k_m) \in r$.
2. If the ground atom is a comparison $k_i\ \theta\ k_j$ then replace the atom in the formula *f* with *true* or *false* as appropriate. □

Example 4.3 Substitution

Consider the following formulas based upon the travel database of Section 3:

f = sites(SNAME, REGION)

g = ∃ADDRESS (hr(HNAME, REGION, ADDRESS) ∧ sites(SNAME, REGION))

Suppose that the domain of tourist attractions included :

{'Imperial Palace', 'Yanaka', 'Fanueil Hall', 'Revere House', 'Tower of London'}

and that the domain of regions included:

{'North End', 'Beacon Hill', 'Hibiya', 'Asakusa', 'Ueno'}

then the following substitutions would be:

f('Imperial Palace'/SNAME, 'Beacon Hill'/REGION) = false

f('Revere House'/SNAME, 'North End'/REGION) = false[10]

g('Dai-Ichi'/HNAME, 'Yanaka'/SNAME, 'Ueno'/REGION) =

∃ADDRESS (hr('Dai-Ichi', 'Ueno', ADDRESS) ∧ *true*). □

Definition 4.4 Interpretation

The interpretation of a formula *f* with no free domain variables, written *I*(*f*), is recursively defined as:

1. If *f* is *true* or *false* then *I*(*f*) is *true* or *false*.
2. If *f* is ¬ *g* and *g* has no free variables, we say if *I*(*g*) is *true*, *I*(*f*) is *false*. Otherwise, *I*(*f*) is *false*.
3. If *f* is *g* ∧ *h* then *I*(*f*) is *true* when both *I*(*g*) and *I*(*h*) are *true* and *false* otherwise.
4. If *f* is *g* ∨ *h* then *I*(*f*) is *false* when both *I*(*g*) and *I*(*h*) are *false* and *true* otherwise.
5. If *f* is ∃*X*(*A*)*g* where only *X* is free in *g*, then *I*(*f*) is true when there is at least one value $c_i \in$ dom(*A*) for which *I*(*g*(*c/X*)) = true.
6. If *f* is ∀*X*(*g*) where only *X* is free in *g*, then *I*(*f*) is true when for every value $c_i \in$ dom(*A*) for which *I*(*g*(*c/X*)) = true.
7. If *f* is (*g*) then *I*(*f*) = *I*(*g*). □

[10] note that the Revere House may indeed be in the North End, but this is not a fact in the relation *sites*. Therefore the predicate evaluates to *false*.

Definition 4.5 Domain relational calculus (DRC) expression
A calculus *expression* has the form $\{X_1,X_2,...,X_n \mid f(X_1,X_2,...,X_n)\}$ where, as indicated above, $f(X_1,X_2,...,X_n)$ is a legal calculus formula and $X_1,X_2,...,X_n$ corresponding to attributes $A_1,A_2, ... ,A_n$ are the only free domain variables in *f*. The value of an expression *E* on database *d* is therefore a relation *r* having scheme (*J,D*) for tuples of the form $(A_1,A_2, ... ,A_n)$ and containing all tuples $(c_1,c_2,...,c_n)$ where $c_i \in D_i$ and $I(f(c_1/X_1,c_2/X_2,..., c_n/X_n)) = true$. □

Example 4.4 A query as a domain relational calculus expression
We can translate the query which regions have a station or tourist attraction? Into the following expression:
{REGION | ∃SNAME, STATION (sites(SNAME, REGION) ∨ trains(STATION, REGION))} □

A domain calculus expression is therefore merely one way of articulating a query over a set of base relations. The expression $\{X_1,X_2,...,X_n \mid f(X_1,X_2,...,X_n)\}$ is a query for all tuples in the database that satisfy the query constraints in *f*. The answer to the query is a relation whose schema is $(A_1,A_2, ... ,A_n)$.

There is one significant problem with this definition of expressions and interpretations. Relations are defined as finite subsets of the Cartesian product of the domains $D_1 \times D_2 \times \ldots \times D_n$. However, domains themselves could be infinite.[11] The problem arises when we attempt to find an interpretation for legal calculus expressions that query infinite domains, possibly producing infinite relations.

Example 4.5 Negation and infinite relations[12]
In Q8 of Section 3, we asked, "List all hotels that are <u>not</u> in the same region as the *Imperial Palace* in Tokyo, Japan." Knowing that the Imperial Palace is in the Hibiya region of Tokyo, consider a simpler variant on this query which asks "List all hotels that are not in the Hibiya region of Tokyo." We might translate this query into the following calculus expression:

{HNAME |¬regions(HNAME,'Hibiya')}.[13]

[11] Consider the domain for the attribute price from our earlier examples. We certainly would not want the database to set an arbitrary bound on the maximum price a hotel could charge for one night's stay. Likewise, the domain for the attribute name might include hotels from around the world including "Le Meridien, Boston" in Boston, MA and the "Warwick Hotel" in Philadelphia, PA. The Cartesian product of name and price includes all possible permutations of hotels worldwide and an infinte range of prices. However, as indicated in our example database from Section 3, the relation hotels contains a finite subset of hotel names corresponding to establishments in Tokyo, Japan and only the corresponding prices charged by those Tokyo hotels.

[12] We use the example of negation here, but similar problems exist for interpreting existential and universal quantification. See [Maier, 1983 #16 at 244-49.

[13] {name |¬regions(name,"Hibiya")} is shorthand. The formal expression would be {name|∃ region (¬ (regions(name,region)) ∧ (region = "Hibiya"))}. Because "Hibiya" is the only possible substitution for region, we remove the existential quantifier and substitute "Hibiya" as a constant in the remaining expression.

The answer, of course, is the presumably rather large set of substitutions for HNAME, (c/name) such that (c, 'Hibiya') is not a pair in table regions. □

To address the problem of infinite relations, we reach beyond the predicate calculus framework to further restrict the types of expressions that we consider evaluable. This concept is called *safety*.[14] The general intuition behind safe calculus expressions is that interpretation of domain variables somehow be explicitly constrained to some finite set of values. To guarantee this, we claim that value(s) which make the expression true must come from a domain consisting of all values that appear either in the constants or in the (finite) relations mentioned in the query.[15]

Example 4.6 Limited expressions
Rewrite the previous example to "bind" to a finite domain. In this case, we use Hibiya:
{NAME | ∃REGION(regions(NAME, REGION) ∧ ¬(REGION = 'Hibiya'))}

Both the quantified variable region and the free variable name are limited by relation regions. In evaluating the existential quantifier, although the domain for variable region might be infinite, we need only consider values that appear in regions. Likewise, we only consider possible names that appear in regions. □

Definition 4.6 Safe DRC
Formally, we define the construction of a *safe* DRC formula following Ullman (1988; 1989) as one where every free variable must appear in at least one non-negated atomic formula that corresponds to a finite relation.

1. There are no uses of universal quantification.
2. Disjuncts must have the same set of free variables.
3. For a maximal subformula that is the conjunction of one or more formulas $F_1 \wedge F_2 \wedge \ldots \wedge F_m$, variables free in any F_i must be *limited* such that
   3.1. A variable is limited if it is free in a formula F_i that is not a comparison and is not negated.
   3.2. If F_i is a comparison $X=c$ then X is limited.
   3.3. If F_i is a comparison $X=Y$ and Y is limited then X is limited.
4. A negated formula is unsafe **unless** it appears in a disjunct with one or more non-negated conjuncts and the free variables in the negated formula are limited as per rule 3. □

Fortunately, it turns out that these limitations do not compromise the expressiveness of our queries with respect to the algebraic relational operators with which users are typically aware and which we use as a reference (relational completeness).[16] Consequently further references to the DRC will refer to the safe-DRC unless explicitly noted otherwise. In particular, we will use the DRC and the value of an expression to formally define our concept of attribution.

[14] There is a more general notion of safety that does not contribute to our definition of attribution and so is overlooked. See "limited evaluation" in [Maier, 1983 #16]or "domain independence" in (Ullman 1988).
[15] (Maier 1983)
[16] See (Ullman 1988 at 153) for a proof on the equivalence of the safe DRC and the relational algebra.

4.2 Attribution and the DRC

We initially suggested that the intuition for attribution was somehow related to an interpretation for the logical expression of a query. We can now be slightly more specific about that idea. A relation *r* is the result of a query *Q* denoted by the DRC expression $\{X_1,X_2,\dots,X_n \,|\, f(X_1,X_2,\dots,X_n)\}$. The attribution of the tuples $(c_1,c_2,\dots,c_n)$ of *r* when *Q* is evaluated on database *d*, denoted $Attr(r, (c_1,c_2,\dots,c_n), Q, d)$ is related to the set of substitutions $f(c_1/X_1,c_2/X_2,\dots,c_n/X_n)$ such that $I(f(c_1/X_1,c_2/X_2,\dots,c_n/X_n)) = true$.

This must seem rather tautological. The attribution of a relation is somehow the relation itself. Therefore, we develop the idea by first considering attribution for *conjunctive queries* and then iteratively refining the model over progressively more general classes of queries.

4.3 Conjunctive queries

We begin the construction of our attribution model by first limiting the range of possible queries to the class of conjunctive queries (CQ). Were we to limit our model to conjunctive queries, attribution would still prove quite useful, for we know that CQ correspond to the class of all SQL queries constructed using selection-on-equality, project, and natural join (Maier 1983; Ullman 1988).

We define three different types of attribution for CQ expressions. After providing a definition for attribution equivalence, we confirm the equivalence of the attribution for equivalent CQ expressions. An algorithm for composing the attribution of an expression by iteratively drilling down through IDB is presented. We verify that composition produces the same attribution as the equivalent, unified query expressed only on EDB. Finally, we present some remarks on attribution granularity. We note the parallel between attributing some subset of values in a result and attributing using only some subset of values in the input sources.

4.3.1 Attribution concept

We first define the term *conjunctive query* and then develop our model by considering our original intuition for attribution as a set of substitution lists for the variables in the expression.

Definition 4.7 Conjunctive query
A *conjunctive query* is an expression of the form:

$$\{X_1,X_2,\dots,X_n \mid \exists Y_1,Y_2,\dots,Y_m\, f(X\, X_1,X_2,\dots,X_n,Y_1,Y_2,\dots,Y_m)\}$$

constructed from a subset of the DRC, as defined earlier, consisting only of domain variables, constants, predicates that represent relations, conjunction, and existential quantification.[17] □

[17] Because we can rewrite $r(X_1,X_2,\dots c\dots,X_n)$ as the formula $(r(X_1,X_2,\dots Y\dots,X_n) \wedge (Y = c))$ we see that conjunctive queries permit a safe or limited form of equality through multiple occurrences of the same variable in multiple conjuncts. See (Ullman 1988).

Example 4.7 Conjunctive queries
Conjunctive queries from the examples in Section 3 are:

$E_1 =$ {HNAME | hotels(HNAME, ROOMS, PRICE)}
$E_2 =$ {HNAME | hostels(HNAME, PRICE, STATION) $\wedge$ sites("Nakamise Dorsi", STATION)}
$E_3 =$ {HNAME | regions(HNAME, REGION) $\wedge$ sites("Imperial Palace", REGION) $\wedge$ hotels(HNAME, ROOMS, PRICE)} □

If attribution consists of the set of substitutions for all variables in the expression, as demonstrated in Example 4.3, then attribution appears to combine a number of distinct concepts together at once. For example, there are a number of relations and variables used to determine the result that are not reflected in the set of free variables. Specifically, how do we know that the Imperial Palace and the hotels in our answer are in the same region? We need to know what region the Imperial Palace is in and what region each of the hotels are in. More generally, distinct information is conveyed in various subsets of the free and bound variables.

4.3.2 Types of attribution

Combinations of free and bound variables in the expression correspond to the intuition introduced in Section 3 that there are different types of attribution depending upon a particular user's interest. In this thesis, we will address three distinct subsets of the set of all variables and constants in an expression.

Perhaps the simplest attribution is that which we demonstrated in Section 4.3.1. From an intellectual property or remuneration perspective, knowing all of the values and variables used, irrespective of the role they play in answering the query, is significant.

Definition 4.8 Comprehensive attribution
The *comprehensive attribution* for the relation represented by a CQ expression $r = \{X_1,X_2,...,X_n \mid (\exists Y_1,Y_2,\ldots,Y_m)\ f(X_1,X_2,...,X_n,Y_1,Y_{2,...,}Y_m)\}$ is a set of pairs where each pair is a substitution list *a* for all of the variables in *f* that make *f* true, and the formula itself. We will sometimes write this as $\{f(a)\}$ or where p_i is a predicate in *f* and c_i is a constant, we might write $\{<p_i(c_i)>\}$ □

Note that for CQ expressions, a minimal list of substitutions must interpret every predicate in the expression as *true*. For an expression with $m + n$ variables, the substitution list must have $m + n$ substitutions.

For E1 in Example 4.7, a substitution *a* in the *comprehensive attribution* will provide values for the variables, HNAME, ROOMS, and PRICE. In addition to identifying all sources consulted in the query, both a unique substitution list and the set of lists convey additional information. In distinct substitution lists for CQ expressions, the same variable can recur in multiple predicates of the same formula. Multiple predicates correspond to multiple sources as in the case of an attribute used in a natural join. Note also that two distinct substitution lists might have the same values for all free variables X_i and differ in at most one existentially quantified variable Y_j hinting at the issue of multiple derivations raised in Section 3. From Example 4.7 we see that each answer in the result is attributable to two distinct substitutions. We will say more about multiple sources below.

A second type of attribution focuses on only the free variables in an expression rather than the set of all variables. Every occurrence of a free variable X_i in a distinct predicate *p* of a CQ corresponds to a *source* for X_i.

Definition 4.9 Source attribution
The *source attribution* for the relation represented by a CQ expression $r = \{X_1, X_2, \ldots, X_n \mid (\exists Y_1, Y_2, \ldots, Y_m)\ f(X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m)\}$ is the set of pairs where each pair is a substitution list *a* for all variables in predicates of *f* that contain free variables and make *f* true, and the formula itself. □

A user interested in data quality characteristics of the answer that depend upon the sources from which the values in the answer are drawn, such as timeliness or accuracy, will examine the *source attribution* for the query result.

A third type of attribution concerns *relevant* sources. The quality of an answer to a query might depend not only upon values reflected in the result but also upon values used in evaluating query (restriction) conditions. We referred to this distinction in Section 3 as the difference between the quality of the answer to the query and the quality of a value in the answer.

The general intuition behind *relevant* substitutions is that omitting or changing one of these substitutions could increase or decrease the subset of domain values for any given free variable, corresponding to an attribute in the result.[18] In Example 4.7, were we to alter the condition "SNAME = 'Imperial Palace'" the query result would certainly differ.

Definition 4.10 Relevant attribution
The *relevant attribution* for the relation represented by a CQ expression $r = \{X_1, X_2, \ldots, X_n \mid (\exists Y_1, Y_2, \ldots, Y_m)\ f(X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m)\}$ is the set of pairs where each pair is the formula *f* and a substitution list *a* for all *relevant* variables in *f* that make *f* true. We use the term *relevant* to capture constraints on the attribute domains represented by the variables in the head of the expression. All variables in the head (free in the formula for the expression) are relevant. In addition, a bound variable is relevant to the result if renaming the variable to some name not already in the expression (or eliminating a constant) would relax a constraint on one or more of the attribute domains in the result relation (free in the formula for the expression). □

[18] Note explicitly the distinction between restriction conditions and existence conditions represented by Cartesian product. We say more about this in the discussion on *result granularity* below.

Example 4.8
Consider again the CQ expressions from Example 4.7

E_1 = {HNAME | hotels(HNAME, ROOMS, PRICE)}
E_2 = {HNAME | hostels(HNAME, PRICE, STATION) $\wedge$ sites("Nakamise Dorsi", STATION)}
E_3 = {HNAME | regions(HNAME, REGION) $\wedge$ sites("Imperial Palace", REGION) $\wedge$ hotels(HNAME, ROOMS, PRICE)}

In addition, consider the more general CQ expression with domain variables as follows.

E_4 = {A | p(A,B,C) $\wedge$ q(C,D,D) $\wedge$ r(F,G,H) $\wedge$ s(H,J,J)}

Only HNAME is relevant in E_1. HNAME, a free variable, is relevant in E_2. STATION is also relevant in E_2. If we renamed the instance of STATION in relation sites, our new expression might appear as E_2 = {HNAME | hostels(HNAME, PRICE, STATION) $\wedge$ sites("Nakamise Dorsi", STATION2)} and the join condition would no longer constrain the possible values of HNAME. Likewise, the substitution "Nakamise Dorsi"/SNAME constrains values of HNAME by placing a bound on the values for STATION which in turn restrict HNAME. Only ROOMS and PRICE are not relevant in E_3. The quality of each answer in E_3 depends upon our knowledge of where the Imperial Palace is located in relation to each of the different hotels. In E_4, neither domain variable J nor domain variable H are relevant. Renaming one instance of H would alter the join condition between the relational predicates r and s. However, while these predicates pose an existence constraint on a tuple of the result set, they do not constrain the domain (and by extension, the quality) of values in the result set. □

We have defined *attribution* to provide variants on the different sources used to evaluate the answer to a query. However, there is often more than one way to ask a question. Likewise, there are often different ways of answering the same question. In the next subsection, we consider *multiple derivations.*

4.3.3 Multiple derivations – the concept

The concept of multiple derivations addresses the observation that we can arrive at the same answer for the same question in different ways. First, we might ask the same question in different ways (equivalent queries). Second, a single query can produce identical answers in different ways.

Assuming the standard, containment-based definition for equivalent queries (Ullman 1989), we further divide the expression of equivalent query expressions into two categories: queries defined on a database comprised of base tables (Extension Data Bases EDB) and queries that also make use of relations defined in terms of other relations (Intensional Data Base IDB) (Ullman 1988). We refer to these as strict equivalence and composition respectively.

We call equivalent expressions defined on the same, extensional database strict equivalents. We first saw an example of strict equivalence in Example 3.5 of Section 3. Now, we adopt a more abstract representation to help generalize the concept.

Example 4.9 Strict Equivalence
Consider the following CQ expressions:

$E_5 = \{X \mid p(X,Y,Z) \wedge p(U,V,W)\}$
$E_6 = \{Z \mid p(Z,Y,U)\}$

Expressions E_5 and E_6 are syntactically different, yet they are equivalent. □

For equivalent queries defined using both intentional and extensional relations, we use the term composition. We first saw an example of Composition in Example 3.9 of Section 3. Now, we provide a more abstract representation.

Example 4.10 Composition
Consider the following CQ expressions:
Assume relation $s(U,V,W) \overset{def}{=} \{U,V,W \mid p(U,V,X) \wedge q(W,X,Y)\}$
and assume relation $E_7 = \{S,V \mid s(U,V,W) \wedge r(S,T,U)\}$.
We can then find a unifier such that $E_7' = \{S,V \mid p(U,V,X) \wedge q(W,X,Y) \wedge r(S,T,U)\}$. □

While equivalent queries lead to identical results, we might also think of a single expression producing identical results. For conjunctive queries, consider the same value appearing in different predicates as in the case of natural join. Natural joins in conjunctive queries were introduced in Q6 of Example 3.12 in Section 3. Here, we again offer a more abstract representation.

Example 4.11 Natural joins

$E_8 = \{X \mid p(V,W,X) \wedge q(X,Y,Z)\}$

Any substitution for the formula in E_8 must include *c/X*, suggesting that the relations represented by predicates *p* and *q* are both sources for *c*. □

Within a single relation we might also think of different sources when we consider non-key values recurring in multiple tuples. We spoke of weak equivalence and defined weak duplicates in Example 3.11 of Section 3. More generally, consider:

Example 4.12 Multiple instances of the same value
$E_9 = \{Y \mid p(X,Y,Z)\}$ where we assume no functional dependencies (or only the trivial dependency where XYZ → XYZ). Then, there may well be multiple values of Y corresponding to multiple tuples <X,Y,Z> in predicate p. □

In describing the issue of multiple derivations for an answer from the same query expression, we allude to the idea that a variable substitution might apply in more than one predicate, and that a single predicate may have duplicate, non-key values. Both issues suggest that there may be more to granularity than a list of substitutions in the formula for a query expression. We may be interested in a specific value (join-attribute or non-key value), and we may wish to distinguish between different substitutions in same attribution set. Issues of result and source granularity are addressed beginning in Section 4.3.7.

4.3.4 Multiple derivations from different expressions – strict equivalence

Our general intuition for attribution equivalence is that the substitutions are the same. In other words, equivalent comprehensive attributions should provide the same interpretation for the same expressions. Source and relevant attributions should be equally comparable. In the case of strict equivalence, if two conjunctive queries E_1 and E_2 are equivalent, then there is a containment mapping from E_1 to E_2 and from E_2 to E_1 (Ullman 1989). These containment mappings map predicates and variables between E_1 and E_2 and satisfy our intuitions about equivalent comprehensive, source, and relevant attributions. We therefore conclude that under different types of attribution, the attribution of equivalent CQ-expressions are equivalent.

First, we provide more formal definitions for what is meant by [comprehensive | source | relevant] attribution equivalence.

Definition 4.11 Attribution equivalence
Two attributions A_1 and A_2 are equivalent when there is a mapping for every variable and its corresponding predicates from A_1 to A_2 and from A_2 to A_1. □

Example 4.13 Attribution equivalence
Consider again Example 4.9.

$E_5 = \{X \mid p(X,Y,Z) \wedge p(U,V,W)\}$
$E_6 = \{Z \mid p(Z,Y,U)\}$

We say that the comprehensive attribution $A_{C5} \equiv A_{C6}$ because the containment mapping from E_5 to E_6 and vice versa, establishing the equivalence of E_5 and E_6, also maps the attribution substitutions.

The mapping establishing the equivalence of source attribution $A_{S5} \equiv A_{S6}$ is just the containment mapping for the free variables in E_5 and E_6. Likewise for the equivalence of relevant attribution $A_{R5} \equiv A_{R6}$. The mapping indicates that there is no free variable for a relational predicate p in E_5, that is not mapped in E_6. This will cause a problem once we add the union operator ($\cup$) into the query language. □

Given our definitions of equivalence, we then propose

Theorem 4.1 Attribution equivalence
If E_1 and E_2 are equivalent CQ expressions, then their [comprehensive | source] attributions, A_1 and A_2, are equivalent. If E_1 and E_2 are minimal, then attribution equivalence holds trivially for comprehensive, source, and relevant attribution.

Lemma 4.1 Comprehensive attributions of equivalent CQ expressions are equivalent.
This is trivially true by the definition of equivalence between E_1 and E_2.

Lemma 4.2 Source attributions of equivalent CQ expressions are equivalent.
Because the queries are equivalent, we know that the two expressions define the same relation. Therefore, in a CQ expression, the mapping must take the predicates containing free variables in E_1 to the predicates containing free variables in E_2 and vice versa.

Lemma 4.3 Relevant attributions of minimal, equivalent CQ expressions are equivalent.
We know that a mapping h from relevant variables in E_1 to variables in E_2 exists by equivalence. We need to verify that h maps all relevant variables in E_1 to relevant variables in E_2 and vice versa. From our definition of relevance, we know that we can exclude any redundant relational predicate as inherently irrelevant. Moreover, we know, from the query optimization literature, that removing redundant predicates from equivalent CQ expressions results in a unique, minimal equivalent CQ expression (Ullman 1989). As a consequence, the relevant attribution of equivalent CQ expressions is trivially equivalent because they are the same. Note that this claim assumes the absence of functional dependencies in the relation. If, for example, a relation has two disjoint candidate keys, then an expression that constrains one candidate key could be equivalent to an expression that constraints the second candidate key.

By Lemmas 4.1, 4.2, and 4.3, we conclude that the comprehensive and source attributions of equivalent queries is equivalent while the attributions of the minimal equivalents of equivalent queries are identical. □

4.3.5 Multiple derivations from different expressions, composition

A second way in which we get different expressions for the same query is when some predicates are defined in terms of others. As seen in Section 4.3.3, when we allow intentional databases (IDB), equivalent CQ-expressions can introduce new predicates and variables. We define the attribution of an expression involving IDB by rewriting the expression in terms only of the base data sources following the process of Unification in datalog queries (Ullman 1988).

The principle of composition establishes that, instead of re-writing the query, we may determine the attribution for composed queries in a recursive manner. First determine the attribution A in terms of both EDB and IDB. Extend each substitution $a_i \in A$ as follows. Treat every reference to an IDB as an independent CQ expression; extend a_i by attributing each IDB. For successive unfoldings, assuming that no recursive definitions are allowed, we eventually arrive at the attribution for the initial expression in terms of base data sources.

Example 4.14 Attribution composition

$$E_1 = p \wedge q \wedge r$$
$$r \stackrel{\text{def}}{=} E_2 = s \wedge t$$
$$E_3 = p \wedge q \wedge s \wedge t$$

Step 1. Get the attribution for E_1 in terms of p, q, and r.

Step 2. Project the substitution list from Step 1. onto *r*.
Step 3. Attribute Step 2. on the expression for *r*.
Step 4. Combine the attribution from Step 3. to the attribution from Step 1. □

More generally, we propose a CQ expression *E'* for a database *d* of the form:

$$p_1 \wedge p_2 \wedge \ldots \wedge p_n \wedge q_1 \wedge q_2 \wedge \ldots \wedge q_m$$

where p_i is a predicate for a relation $r_i \in d$ and $\forall j$, q_j is a predicate for a relation $r_j \notin d$ and q_j is defined by a CQ-expression over predicates p_i.

Definition 4.12 Attribution of a composed expression
The attribution of the result *r* from *E'* defined on *d'* in terms only of relations in *d*, is defined as *attr*(*r*, *E*, *d*) where *d* explicitly excludes $\forall j$, predicates q_j and *E* is the re-write of *E'* in terms of *d*. □

It follows that we can build progressively deeper layers of indirection by defining a set of predicates r_k defined in terms of p_i's and q_j's and so forth resulting in correspondingly more complex re-writes.

While re-writing provides us with a consistent definition for the attribution of expressions in the presence of views and base relations, it presents some pragmatic challenges. Neither user nor system may initially be aware of underlying data sources. Users may be uninterested in pursuing the attribution of certain intermediate-level sources. Rather than re-writing the entire query a priori, we would prefer to attribute by iteratively unfolding successive layers of IDB as necessary.

Algorithm 4.1 Attribution composition

```
Compose ($A$,$f$) {
if $f$ has no $q$'s then return $A$
else pick $q_i$, an IDB in $f$
    $f := p_1 \wedge p_2 \wedge \ldots \wedge q_{i-1} \wedge q_{i+1} \ldots q_m$
    Compose (Unfold ($A$, $q_i$),$f$) }

Unfold ($A$, $q$) {
if $A$ is $\varnothing$ then return { }
else pick $(a,f) \in A$
    let $g$ be the formula for IDB $E$ representing $q$
    let $u$ be the unifier for $h = unify(f,g)$
    let $E'$ be $E$ as defined by $g$ with the renaming of $u$
    $B$ = attr( $a(q)/x$, $E'$, $d'$)
    Rewrite ($B$, $u(a - a(q))$, $h$) $\cup$ Unfold ($A - \{(a,f)\}$, $q$) }

Rewrite ($B$, $a$, $h$) {
```

```
if $B$ is $\varnothing$ then return { }                          (15)
else pick $(b,g) \in B$                                          (16)
    $\{<\{a \circ b\},h>\} \cup$ Rewrite $(B - \{(b,g)\}, a, h)$       (17)        □
```

We use Compose (A, f) to recurse through the IDB in f. For each IDB, we find the definition for the IDB in line (9) and find a unifier in line (10) to be certain that we can rename variables appropriately. In line (11), we rewrite the expression for the IDB accounting for the variable renaming and call this E'. Finally, we attribute the specific tuple in the IDB by pushing constants from the original substitution into the corresponding variables of E'. We denote this as $E'(a(q)/x)$ in line (12). Because this attribution itself returns a set of substitution - formula pairs, we replace the original substitution in A with the set of substitutions from the attribution of E'. Note in line (17) where the set of new pairs uses the unified formula h and combines the original substitution a with substitutions for the IDB b. Line (13) simply removes the duplicate substitutions.

Theorem 4.2 Attribution composition
Attribution composition computes the attribution of a composed expression.

Assume without loss of generality the following CQ expressions E_1, E_2, E_3 defined by the formulas f, g, and h respectively s.t.
$E_1 \overset{\text{def}}{=} f = (p_1 \wedge p_2 \wedge \ldots \wedge p_n \wedge q)$ where q is the only IDB in E_1
$E_2 = q \overset{\text{def}}{=} g = (\ r_1 \wedge r_2 \wedge \ldots \wedge r_m)$ where $r_i \in d$
$E_3 \overset{\text{def}}{=} h = (\ p_1 \wedge p_2 \wedge \ldots \wedge p_n \wedge r_1 \wedge r_2 \wedge \ldots \wedge r_m)$

Note that the formula g for E_2 already has variables renamed and reordered as in Line (10) and (11) so that references to E_2 in the proof below correspond to E' in Line (12).

Given E_1 defined on $d' = d \cup \{q\}$ and r, the result of evaluating E_1 on d', attribution composition computes the [comprehensive | source | relevant] attribution of result r in terms of d as defined by attr(r, E_3, d).

Lemma 4.4 $(a_3,h) \in A_3$ is a comprehensive attribution for E_3 if and only if $(a_3,h) \in$ Compose(E_1,f) .

Case (→)
Pick a random substitution $(a_3,h) \in A_3$ and split it: Project a_3 onto f and g.
We know that $a_3(f) = a_1 \in A_1$ because $\forall i,\ I(p_i(a_3(p_i)/x)) = true$ and $\forall j,\ I(r_j(a_3(r_j)/x)) = true$ where q is defined by the r_j's. Similarly, we know that $a_3(g) = a_2 \in A_2' \subseteq A_2$ where A_2' as the attribution for the tuple defined by $a_1 \cap a_2$, a tuple in q.
Compose passes A_1 to **Unfold**. **Unfold** calls attr($\{a_3(q)\}$, E_2, d') which looks for substitutions of E_2 with $a_3(q)$ pushed into the expression. Attr($\{a_3(q)\}$, E_2, d') is $A_2' \subseteq A_2$ because the attribution of $E_2 = A_2$ and $a_1(g)$ makes E_2 true therefore $A_2' \subseteq A_2$.

Unfold is applied to every value of A_1 so certainly it calls itself on a_1.
Unfold calls **Rewrite** with a_1 and A_2'.
Rewrite is applied to every element of A_2' so certainly is is applied to a_2.
But **Rewrite** takes h, the unification of f and g, and returns $a_1 \circ a_2$ which is a_3.
Case (←)
If (unify(f,g)) = h, does every pair ($a_1 \circ a_2$, h) appear as a substitution in A_3? Pick some arbitrary a_1 from a pair in A_1. Now we cannot pick just any a_2. **Compose** creates A_2' from attr($\{a_1(g)\}$, E_2, d'). So pick any a_2 from a pair $\in A_2'$. We know $a_1 \circ a_2$ paired with h appears in A_3 if it makes E_3 true. $\forall i, I(p_i(a_1(p_i)/x)) = true$ and $\forall j, I(r_j(a_2(r_j)/x)) = true$. But are E_1 and E_2 true at the same time (i.e. do they make h true)? Because we know a_2 is from a pair in A_2' by construction, we know that a_2 makes E_2 true for a true interpretation of A_1. Therefore, we know that ($a_1 \circ a_2$, h) $\in A_3$. □

Lemma 4.5 (a_3,h) $\in A_3$ is a source attribution for E_3 if and only if (a_3,h) $\in$ Compose (A_1,f) where A_1 is the source attribution for E_1.

Case (→)
Pick a random source attribution (a_3,h) $\in A_3$ and split it: Project a_3 onto f and g. These are just the free variables in E_3 and accompanying variables that identify unique instances of tuples containing a particular value for a free variable. We know as before that $a_3(f) = (a_1,f) \in A_1$ because for the predicates p_i, $E_1 \subseteq E_3$ and for predicate q, because q is defined in terms of the predicates r_j, we know that the free variables for q are also assigned in $a_3(f)$. Similarly, we know again that $a_3(g) = (a_2,g) \in A_2' \subseteq A_2$ where A_2' as the attribution for the tuple defined by $a_1 \cap a_2$, a tuple in q.
Compose passes A_1 to **Unfold** with $q \overset{\text{def}}{=} E_2$ with formula g.
Unfold is called on every value of A_1 so certainly it is called on a_1.
Unfold calls attr($\{a_1(g)\}$, E_2, d') which we know is $A_2' \subseteq A_2$ because the attribution of $E_2 = A_2$ and $a_1(g)$ makes E_2 true therefore $A_2' \subseteq A_2$.
Rewrite is called for every element of A_2' so certainly it is called for a_2.
But **Rewrite** takes h, the unification of f and g and returns $a_1 \circ a_2$ which is a_3.

Case (←)
If (unify(f,g)) = h, Does every pair ($a_1 \circ a_2$, h) appear as a substitution to a relational predicate containing a free variable in A_3? Pick some arbitrary a_1 from a pair in A_1. Now we cannot pick just any a_2. **Compose** creates A_2' = attr($\{a_1(g)\}$, E_2, d'). So pick any a_2 from a pair $\in A$-$_2'$. We know $a_1 \circ a_2$ paired with h appears in A_3 if it makes E_3 true. $\forall i, I(p_i(a_1(p_i)/x)) = true$ and $\forall j, I(r_j(a_2(r_j)/x)) = true$. But are E_1 and E_2 true at the same time (i.e. do they make h true)? Because we know a_2 is from a pair in A_2' by construction, we know that a_2 makes E_2 true for a true interpretation of A_1. Therefore, we know that ($a_1 \circ a_2$, h) $\in A_3$. □

Lemma 4.6 $(a_3,h) \in A_3$ is a relevant attribution for E_3 if and only if $(a_3,h) \in$ Compose (A_1,f). Where A_1 is the relevant attribution for E_1.

This is more complicated because we need to verify that $relevant(E_3) = relevant(E_1) + relevant(E_2)'$ where $relevant(E_2)' \subseteq relevant(E_2)$ and $relevant(E)$ refers to the relevant variables in E and likewise for $free(E)$; $bound(E)$. We form $relevant(E_2)'$ as we formed A_2' previously. We attribute only the relevant variables in q on the expression E_2. For convenience, we assume that the CQ expression is minimal.

Case (→)
Pick some relevant attribution $(a_3,h) \in A_3$ and split it: Project a_3 onto f and g.
We need to establish that $a_3(f) = (a_1,f) \in A_1$ and $a_3(g) = (a_2,g) \in A_2' \subseteq A_2$ where A_2' as the attribution for the tuple defined by $a_1 \cap a_2$, a tuple in q.
A substitution c/X is in a substitution list for a_3 because either X is free in E_3 or c/X joins two relational predicates, at least one of which is recursively joined to a relational predicate containing a free variable or is a constant from the original query expression that appears in a relational predicate recursively joined to a predicate containing a free variable of E_3.

Case 1. $X \in relevant(E_3)$ and $X \in free(E_3)$. $Free(E_3) = X \in free(E_1) \subseteq relevant(E_1)$ by definition of the equivalence of E_1 and E_3.
for $X \in free(E_3) = Y \in free(E_2)$, Y must also be free in E_1 because E_2 is q in E_1 (e.g. $Y \in free(E_2) \rightarrow Y \in free(E_1)$). Consequently, at least for the relevant variables in E_2 that are free, we know $a_2 \in A_2' \subseteq A_2$

Case 2. $X \in relevant(E_3)$ joins relational predicates to a recursively joined set of relational predicates or X constrains one predicate in a recursively joined set of relational predicates (e.g. X is a constant or X appears multiple times in a single relation). All such predicates are in the set p_i and at least one joined predicate contains a free variable in h. Then X is relevant in E_1 so $a_3(f) = a_1$ for $(a_1,f) \in A_1$.

Case 3. $X \in relevant(E_3)$ is like Case 2 except all such predicates are in the set r_j. Then X is relevant in E_2 so $a_3(g) = (a_2,g) \in A_2' \subseteq A_2$ where A_2' as the attribution for the tuple defined by $a_1 \cap a_2$, a tuple in q. (Recall that $g \overset{\text{def}}{=} q$ in E_1).

Case 4. $X \in relevant(E_3)$ appears in both some predicate p_i and some predicate r_j. Then X must appear in predicate q of E_1 (X is not free in E_1 so it must be bound in E_1 and appear in q in order to appear in both p_i and r_j in E_3). Therefore X is relevant in E_1 so $a_3(f) = a_1$ for $(a_1,f) \in A_1$.

It is possible that $a_3(g)$ is empty, which occurs when we consider a Cartesian Product and then do not restrict variables between arguments to the Cartesian Product. In this case, attribution relevance is trivially true in the p_i's.

From here, we do the same unfolding as before and conclude that **Compose** returns $a_1 \circ a_2$ which is a_3.

Case (←)
Pick some random substitution list $a_1 \circ a_2$ as before and verify that $(a_1 \circ a_2,h) \in A_3$.
Proof by contradiction. Suppose not. Then there must be a substitution $c/X \in a_1 \circ a_2, c/X \notin a_3$, or $c/Y \in a_3, c/Y \notin a_1 \circ a_2$.

Case 1. Pick Y. If Y is relevant in E_3, then Y must constrain the free variables in h in some way.
If Y is a free variable, then $c /Y \in a_1$, a contradiction.
If Y constrains a predicate containing a free variable through some recursively joined set of predicates amongst the p_i's, then $c/Y \in a_1$, a contradiction.
If Y constrains a predicate containing a free variable through some recursively joined set of predicates amongst the r_j's and is $a_1 \circ a_2$ as assumed above, then $c/Y \in a_2$, a contradiction.
If Y is relevant and appears in both some predicate p and some predicate r then $c/Y \in a_1$, a contradiction (see Case 4 for the (→) direction).
Therefore, by contradiction, we conclude that there is no $c/Y \in a_3, c/Y \notin a_1 \circ a_2$.

Case 2. Pick X.
If $c/X \in a_1$ because it is a free variable in E_1, then by definition, $c/X \in a_3$, a contradiction.
If $c/X \in a_1$ because it constrains a free variable through predicates p_i, then $c/X \in a_3$, a contradiction.
If $c/X \in a_1$ and appears in both in q's and p's, then $c/X \in a_3$, a contradiction.
Now we need to be careful. Remember that a_2 is selected from attributing $a_1(g)$. It is possible for $a_1(g) = \{\}$ as in the case of Cartesian Product. attr($\{a_1(g)\}$, E_2, d') is non-empty only when there is a relevant variable in q.
If $c/X \in a_2$ because it is free in E_2 and free in E_1, then we know $c/X \in a_3$, a contradiction.
If $c/X \in a_2$ because it is free in E_2 and bound and relevant in E_1, then we know $c/X \in a_3$, a contradiction.
If $c/X \in a_2$ because it is bound in E_2, occurs among the predicates r_j and constraints a free variable in E_2 that is relevant in E_1 (through predicate q in E_1), then we know $c/X \in a_3$, a contradiction. □

Therefore, we conclude that attribution composition computes the attribution of a composed expression. □

It is important to note the subtlety required in composing relevant attribution. Our definition of relevance depends upon drawing a distinction between constraints on attribute domains and explicit query syntax. We saw some challenges for managing relevant attribution in Example 3.9 of Section 3. Consider, more generally, two equivalent queries where selections are pushed down in one case but not in the other.

Example 4.15 Composing relevant attribution

$E_{10} = \{A \mid p(ABCDEF) \wedge s(FGH)\}$
$E_{11} = \{A \mid q(ABC) \wedge r(DEF) \wedge s(FGH)\}$
where $p(ABCDEF) = q(ABC) \wedge r(DEF)$

Syntactically, we observe that F is relevant to A in E_{10} but not in E_{11}. Yet, the equivalence of E_{10} and E_{11} confirms that F indeed does not constrain values of A in the result.[19] □

Theorem 4.2 confirms our intuitions about how attribution should work in the context of composed queries. It indicates that, at least for conjunctive queries, we can recursively drill down through progressive layers of indirection. More generally, Theorem 4.1 and Theorem 4.2 together allow us to conclude that, though there are many different ways to construct a CQ expression, comprehensive, source, and relevant attributions for equivalent CQ expressions are equivalent.

4.3.6 Multiple derivations within a single expression

We saw in Section 4.3.3 how different substitutions might correspond to the same values within a single expression. Both multiple occurrences of a single variable and multiple substitutions proving the same result are modeled in a straightforward manner.

Multiple occurrences of a variable between expressions, as in the concept of relevant variables, are consistent with the semantics of algebraic natural join. That a single variable appears as a join attribute suggests that it derives from two or more distinct relations in a single expression. See Example 4.11. We will say more about what it means to derive from a relation rather than from a substitution in our discussion of granularity to follow.

In addition to identifying duplicate values through multiple occurrences of a single variable in an expression, non-key values can repeat in different facts of a single predicate corresponding to different tuples of a single relation as in Example 4.12.

Rather than being problematic, however, we believe that this highlights a benefit of using substitutions to define attribution. Duplicate values suggest an opportunity for users to explicitly identify either a specific instance of a value or all such instances. In the relational data model we know that we can identify specific instances through functional dependencies. That our attribution model draws a distinction between specific instances of a value and all such instances introduces the concept of granularity.

[19] Note that we are essentially saying that composition holds for relevant attribution because we explicitly define composition and relevance that way.

4.3.7 Granularity – the concept

The intuition behind granularity is that attribution is simply a pointer from query results to query sources. Granularity addresses the precision with which the pointer identifies data in a source or in a result. Source granularity allows the user to receive a list of references that provides greater (or less) detail. Note that a substitution, defined as a list of value-substitutions and the formula to which the substitutions are applied, implicitly associates values with one or more relations. As a consequence, rather than a substitution value, we might return the tuple(s) containing a value or even the relation name. Source granularity was first discussed in Examples 3.15 and 3.16 of Section 3. More abstractly, consider:

Example 4.16 Source granularity

$E_{12} = \{A, E, F \mid p(A, B, C) \wedge q(C, D, E) \wedge r(F, G, H)\}$

where the source of interest is represented by predicate $p(A, B, C)$
if a is a substitution list for the formula of E_{12} then $a(p) = <c_1/A, c_2/B, c_3/C>$ and the substitutions make predicate p true. We can think of a specific tuple instance as a source for the evaluation of E_{12}, (c_1, c_2, c_3). At the opposite extreme, we might roll-up all such tuple references by identifying the relation for predicate p as a source. The two poles define a continuum where, using the notation loosely, we can specify some tighter bound on tuples from the base relation that are used to evaluate the result. Consider, for example, $(c_1,\_,\_)$ as the set of all tuples in the relation for predicate a subset of tuples in the relation for predicate p where the value of the first attribute is c_1. □

Similarly, result granularity allows the user to ask attribution questions to varying degrees of specificity. Initially, we assumed that attribution applied to a query result as a whole. Implicitly, however, we accepted the notion that users might have an interest in only one portion of the result. Indeed our algorithm for attribution composition exploits the fact that we can attribute parts of relations. Rather than asking for the attribution of a relation defined by an expression, we may wish to know the attribution for a specific tuple, column, or value. Example 3.17 of Section 3 offered a first example of result granularity.

Example 4.17 Result granularity

Consider

$E_{13} = \{A, B, E \mid p(A, B, C) \wedge q(C, D, E)\}$

Again using the notation loosely, we might demonstrate an interest only in tuples where the value for variable B is c_2 (denoted $(\_, c_2/B,\_)$). For example, all students in a student database who have the last name "Smith." At the extreme, we might wish to attribute only a single, specific tuple $(c_1/A, c_2/B, c_3/C)$. □

We can therefore think of a query result as a relation and the attribution of that result as the corresponding input relations. However, being able to specify different granularities is useful because it enables precision while at the same time introducing possible efficiencies. When we attribute a relation, we do not necessarily know which substitutions correspond to specific values in the relation. Intuitively, every value is the result of distinct substitutions. If such exactitude is not necessary, however, as in the case of the list of references at the end of a text, attributing a group of values to a single list of relation names reduces the amount of necessary attribution metadata.

4.3.8 Source granularity

In source granularity, we vary the precision with which we identify the formula and the one or more corresponding variable substitutions that together define an attribution substitution. We hinted at source granularity when we discussed multiple derivations within a single expression. In particular, a single substitution may occur in multiple predicates. Multiple facts (with the same non-key attribute values) may correspond to a single value substitution.

Our definitions for different types of attribution correspond implicitly to different source granularities. Comprehensive attribution gives the complete list of substitutions for defining one true interpretation of a CQ expression. Source attribution identifies explicit tuples but only in relations from which free variables are drawn. Relevant attribution defines sets of tuples for selected predicates in the expression.

Example 4.18 Source granules and attribution substitutions
Consider again DRC 2.1 from Section 3.

DRC2.1 {HNAME | regions(HNAME, REGION) $\wedge$ sites("Imperial Palace", REGION) $\wedge$ hotels(HNAME, ROOMS, PRICE)}

The comprehensive attribution for the expression is the set of pairs:

{<*f*("Imperial"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME, "single"/ROOMS, 34000/PRICE)>;
<*f*("Imperial"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME, "double"/ROOMS, 39000/PRICE) >;
<*f*("Dai-Ichi"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME, "single"/ROOMS, 10000/PRICE) >;
<*f*("Dai-Ichi"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME, "double"/ROOMS, 80000/PRICE) >}

As illustrated above, projecting a substitution list onto a relational predicate in *f* returns a tuple that appears in the corresponding relation.

By contrast, consider the relevant attribution:

{<*f*("Imperial"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME)>;
<*f*("Dai-Ichi"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME) >}

Projecting one of the substitution lists onto the predicate hotels returns only the substitution "Imperial"/HNAME which we can apply as:

hotels("Imperial"/HNAME, PRICE, ROOMS) and corresponds to two tuples: ("Imperial", "single", 34000) and ("Imperial, "double", 39000) □

Example 4.18 suggests the ambiguity that can occur in attribution where multiple instances of a value in a source may contribute to a single answer. The ambiguity also offers flexibility, however. Individual variable substitutions indicate all occurrences of one or more variables in an expression whereas attributing with source tuples directs the attribution to identify explicit instances. Note that our use of tuple-level source granularity is a proxy for identifying unique instances. Leveraging functional dependencies may provide additional value here. Buneman et al. also hints at the potential of using functional dependencies in attribution and addresses the issue of unique instances for their more general deterministic semistructured data model (Buneman 01).

We note that an arbitrary granule defines a subset of values in a source (or result) relation. Specifying an arbitrary source granule does not imply that all valid substitutions for the expression are contained within the granule. Likewise, not every substitution within a coarse granule of a CQ expression may give a true interpretation for the expression.

Example 4.19 Interpreting source granules in attribution
Consider a variant on DRC 2.1 from Section 3 where we ask for "single" rooms by the "Imperial Palace."

DRC2.1' {HNAME | regions(HNAME, REGION) ∧ sites("Imperial Palace", REGION) ∧ hotels(HNAME, "single", PRICE)}

The comprehensive substitutions are now:

{<*f*("Imperial"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME, "single"/ROOMS, 34000/PRICE)>;
<*f*("Dai-Ichi"/HNAME, "Hibiya"/REGION, "Imperial Palace"/SNAME, "single"/ROOMS, 10000/PRICE) >}

The corresponding source attribution is:

{<*f*("Imperial"/HNAME, "Hibiya"/REGION, "single"/ROOMS, 34000/PRICE)>;
<*f*("Dai-Ichi"/HNAME, "Hibiya"/REGION, "single"/ROOMS, 10000/PRICE) >}

For coarse grained source attribution, we might identify a granule using only the source substitutions:

{<*f*("Imperial"/HNAME)>;
<*f*("Dai-Ichi"/HNAME) >}

Applied to the predicate hotels, we know that the substitution "Imperial"/HNAME corresponds to two tuples:
("Imperial", "single", 34000) and ("Imperial, "double", 39000)

However, the second of the two tuples does not produce a valid interpretation of the original query expression.

Definition 4.13 Source granularity
A source granule on a relation, denoted by predicate p in a CQ expression E, is defined by a CQ expression on predicate p (i.e., it is a view). □

Observation 4.1 Defining a source granule in terms of substitutions
Although an arbitrary source granule need not include all valid tuples for evaluating the truth of an expression, suppose that we have an expression E with attribution A. If we define our source granules using the substitution $a \in A$, we are assured that the source granules will always contain at least those tuples necessary to evaluate the query and produce the result corresponding to the attribution. □

Example 4.20 Defining a source granule in terms of substitutions
Suppose that we had the attribution A for a query expression E with formula f. f includes the relational predicate p such that for some substitution list $a \in A$, $a(p) = c_1, \ldots, c_n$ and c_i/X_i where X_i is a domain variable in p. We can then define a source granule for p as a query expression $\{Y_1, \ldots, Y_m | \, p(Y_1, \ldots, Y_m)\}$ where we substitute c_i/Y_j as appropriate (e.g. where $X_i = Y_j$). The source granule therefore describes p', a tighter bound on p that still is guaranteed to contain at least those tuples that satisfy the original expression E. □

Tuple-level granularity constitutes a value/variable substitution for every argument in a relational predicate and describes a specific instance of a source relation. As noted above, although we define attribution in terms of substitutions, comprehensive and source attribution provide tuple-level granularity. Assuming no functional dependencies, assigning a value to each domain variable in a relation uniquely identifies an instance of the relation. Substitution-level granularity, such as is used in our definition of relevant attribution, implicitly includes every tuple from each constituent base relation that includes a particular attribute-value/domain variable substitution. At the extreme, we can speak of a relation-level source granule as simply a relation name. At the extreme, rather than attributing with specific substitutions, we can simply provide relation names as a proxy for all tuples in the corresponding relation.

In general, tuple-level substitutions are the finest grained (most specific), and relation-level granules are the most coarse, across all attribution types. This says that, where identifying specific values or instances of values is unimportant, we can always attribute with more general relations. For purposes of intellectual property or remuneration, for example, knowing the relation names may be sufficient. Likewise, for data quality purposes, knowing the relation may be enough to convey information about reputability. By contrast, verifying or correcting anomalous values may require finer granularity.

If we limit granules to those defined by substitutions, then we may make the following two observations about the relationship between different levels of source granularity

Observation 4.2 Generalizing from fine- to coarse-grained source granules
Given a set of source [comprehensive | source | relevant] substitutions that constitute a particular degree of specificity, we may always compose a query over the source granules that will contain at least the original substitutions. At the limit, we can always define a source granule that contains the original substitutions as the original base relation(s). □

Observation 4.3 Specializing from coarse- to fine-grained source granules
Assuming a set of [comprehensive | source | relevant] substitutions that constitute a particular degree of specificity, we may always re-attribute the same query expression and query result and return source granules that contain no more than the original set of substitutions. At the limit, we know that the tightest bound is the set of exactly those comprehensive, source, or relevant tuples that evaluate the expression to true. □

Because we define granularity as a composed query on a source predicate *p*, we may also make the following observations about the implications of varying source granularity on other properties of attribution.

Observation 4.4 Attribution composition is preserved
We define source granules in terms of composed queries on the base sources. Source granules therefore implicitly constitute IDB. At the extremes, either a source granule contains exactly those tuples that evaluate the expression to true or it is the identity on the EDB (i.e. relation-level source granularity). We already know that we can compose tuple-level substitutions. At the opposite extreme, if we attribute with a source relation name rather than a set of source substitutions, we know that we can unfold by composing the relation names of the relations used to construct an IDB. □

Observation 4.5 Attribution of strictly equivalent queries is preserved
For relevant attribution, this is again, trivial. There is a unique minimal equivalent; regardless of the source granularity used, the relevant attribution is identical. For comprehensive and source attribution, we may again rely upon the containment map between equivalent expressions. Because the variables map to one another in the same predicates, we are assured that a source granule in one expression, defined as a query composed on a predicate, prescribes the same subset of base relation tuples in the equivalent expression. □

4.3.9 Result granularity

Result granularity stems from two observations. First, from the beginning, we intuited that users may have some interest in greater precision than simply attributing the result of a query. One tuple or even one value may raise particular interest. We refer explicitly to result granularity in our definition of composition. To compose an attribution recursively, we attribute substitutions in a predicate, not the entire relation represented by the predicate.

A second observation motivating result granularity stems from relational closure and the fact that relational query answers can serve as inputs to subsequent queries. As a consequence, source granularity issues like "all occurrences of a value" or "the specific instance of a value" may apply equally to results as well as to sources.

Example 4.21 Result granularity
Consider a variant on DRC 2.1 from Section 3.

DRC2.3 {HNAME, PRICE | regions(HNAME, REGION) ∧ sites("Imperial Palace", REGION) ∧ hotels(HNAME, ROOMS, PRICE)}

We know that the result set includes the "Imperial, 34000" the "Dai-Ichi, 80000" and the "Dai-Ichi, 10000". A user might only have an interest in the "Dai-Ichi" hotel rather than the "Imperial". A different user might only be interested in the attribution for values of PRICE. □

The concept of result granules is consistent with our definitions of attribution, which refer to the substitutions that make the expression for the result true. As with source granules, we can imagine attributing the specific instance of a value in the result rather than all instances of a value. In Section 3 we saw how the projection of a non-key attribute from a base relation can result in multiple sources for the same value.

Mindful that an IDB is simply the result of a query[20], we follow our definition of source granules in defining result granules.

Definition 4.14 Result granularity
A granule of result *r*, defined by CQ expression *E* evaluated on database *d* is a result *r'* defined by a CQ expression *E'* composed on *E* for database *d*.[21] □

Observation 4.6 Attribution of a result tuple
It follows from our definition of a result granule that the attribution of a specific tuple *t* in a result *r*, assuming no knowledge of functional dependencies, is then simply the attribution of composing a query on the result *r* for the specific tuple of interest.

Moreover, because we define result granules using query composition, we are assured of

[20] The closure property of relational theory dictates that a query result (output) may in turn serve as a source (input) to some other expression (Maier 1983; Ullman 1988)

[21] As we enrich our query language, we will eventually define a source or result granule by composing any positive query on a source or result relation, respectively.

Observation 4.7 Attribution of strictly equivalent queries is preserved.
We already know that the comprehensive and source attribution of strictly equivalent queries is equivalent. Because this equivalence is preserved over composition, we conclude that the attribution of an arbitrary result granule is equivalent given equivalent CQ expressions. □

To define the relationship between result granules and source granules we offer the following observations.

Observation 4.8 Attribution of a result tuple
For relation-level source granules, the attribution of one tuple in the result of a CQ expression is the same as any other tuple in the same result. This merely conforms to the intuition that in a CQ expression, every conjunct applies equally to every tuple. □

Observation 4.9 Comprehensive attribution of result values
For comprehensive attribution, we may make the following stronger claims. First, regardless of source granularity, we observe that the comprehensive attribution for one value in a result tuple is the same as that for every other value in the same tuple. Second, if we limit ourselves relation-level source granules, the comprehensive attribution for a value in the result is the same as that for every other value in the result. □

The relationships between different granules has particular relevance for practical implementation, because it promises significant reductions in the amount of attribution metadata necessary to satisfy different user objectives.

4.4 Adding theta comparisons

We now move to refine our theory by extending the richness of the query language. The introduction of theta comparisons challenges some of our earlier conclusions about attribution when limited to CQ expressions. However, we verify that, for strictly equivalent queries, the comprehensive and source attribution of equivalent queries remains equivalent. Moreover, we conclude that for all types of attribution, attribution composition continues to hold.

4.4.1 Attribution concept

The first language extension introduces arithmetic comparisons in atoms of the form ($X\theta Y$), ($X\theta c$), or ($c\theta X$) where c is a constant and X and Y are either free or bound variables that are limited in the manner defined for the DRC above. We refer to our extended queries as CQT expressions (or CQ expressions with theta comparisons). The set of θ operators are {<, ≤, ≥, >, and ≠}. For current purposes, we exclude explicit equality from the set of comparisons; explicit equality is incorporated into the language independently.[22]

[22] Recall that conjunctive queries already included a "safe" or limited version of equality-comparisons. See note and text at 9.

Example 4.22 *θ*-comparison
First, consider a variant on query Q2 of Section 3.

E_{14} = {HNAME, PRICE | regions(HNAME, REGION) $\wedge$ sites("Imperial Palace", REGION) $\wedge$ hotels(HNAME, "single", PRICE) $\wedge$ (PRICE < 15000)}

Second, we present a more abstract case.

E_{15} = {W | p(V,W,X) $\wedge$ q(X,Y,Z) $\wedge$ (V > 10)} □

Extending our definitions of attribution from CQ-expressions, we see that the introduction of comparisons does not introduce new relational predicates but may introduce new variables or perhaps constants for comparison. To better understand the implications of these changes for our theory, we revisit our analysis for conjunctive queries beginning with types of attribution.

4.4.2 Types of attribution

We initially defined comprehensive attribution as a set of substitution lists for all variables in the expression applied to the formula for the expression itself such that the interpretation of the formula is true. The definition for comprehensive attribution remains unchanged.

While θ-comparisons may introduce new variables into the expressions, under the limitations of safety, every variable is still *limited* in the sense that it must appear in a (non-negated) relational predicate. Consistent with Definition 4.9 on source attribution and Definition 4.13 on source granularity, non-predicate atoms are not considered sources. For E_{14} above, the arithmetic comparison is not considered a source for PRICE.

Likewise, Definition 4.10 for relevant attribution remains unchanged. The introduction of comparisons, however, does provide new alternatives for constraining the domain of a free variable. In E_{15}, V is relevant to W.

4.4.3 Multiple derivations from different expressions, strict equivalence

The same two categories for multiple derivations that we identified in CQ expressions apply when theta-comparisons are added. Multiple derivations may stem from equivalent expressions or from multiple occurrences within a single expression. For equivalent expressions on the same database, we now need to consider containment not only between predicates of equivalent expressions but between non-predicate atoms as well.

Example 4.23 Multiple derivations

E_{16} = {XY | p(XYZ) $\wedge$ q(UVW) $\wedge$ (X $\neq$ U) $\wedge$ (X $\leq$ U)}
E_{17} = {XY | p(XYZ) $\wedge$ q(UVW) $\wedge$ (X < U)} □

The problem is tied to the introduction of new atoms in the form of theta comparison. The relationship between arithmetic comparisons of equivalent queries is not always clear as indicated in the following example from Ullman (1989).

Example 4.24 Interactions between arithmetic comparisons and relational predicates

$E_{18} = \{XY \mid p(XYZ) \wedge q(UV) \wedge (U \leq V)\}$

$E_{19} = \{XY \mid p(XYZ) \wedge q(UV) \wedge q(VU)\}$ □

Fortunately, we do know that a containment mapping does hold between the relational predicates in equivalent CQT expressions (Ullman 1989). Furthermore, the property of safety guarantees that all domain variables are captured in the containment mapping.

Theorem 4.3 Attribution equivalence
If E_1 and E_2 are equivalent CQT expressions, then their [comprehensive | source] attributions, A_1 and A_2, are equivalent.

Lemma 4.7 Comprehensive attributions of equivalent CQT expressions are equivalent.

This is trivially true by the definition of equivalence between E_1 and E_2. We know that there is a containment map between all predicates representing relations of equivalent CQT expressions. Moreover, because of safety, we know that the built-in predicates use only variables that are bound in (and hence captured by the containment mapping between) relational atoms.

Lemma 4.8 Source attributions of equivalent CQT expressions are equivalent.

Recall that source attribution is defined in terms of the free variables of a CQT expression. Because the queries are equivalent, we know that the two expressions define the same relation. Therefore, the containment mapping between relational predicates of equivalent expressions must take relational predicates containing free variables in E_1 to the corresponding relational predicates in E_2 and vice versa.

From Lemmas 4.7 and 4.8, we conclude that the comprehensive and source attributions of equivalent queries is equivalent. □

4.4.4 Multiple derivations from different expressions, composition

Composition, our reference for equivalent expressions defined on different databases, does not apply to non-predicate atoms, because theta-comparisons are not defined by expressions. We will, however, want to consider, the effect of non-predicate atoms on our definition for the attribution of composed expressions and whether the theorem for the recursive composition of attribution holds over theta-comparisons.

Again, we rely upon the fact that, though there is no unique, minimal query, there remains a containment mapping between the predicates in equivalent CQtheta queries.

What is the definition of a composed query (e.g. you can substitute expressions with the theta operator in it) and algorithm … do you need to adjust either the drill down or the way you reconstruct the attribution as you back out?

Consequently, the introduction of inequality comparisons does not change the ability to compute attribution in a recursive fashion for predicates composed on other predicates.

Theorem 4.4 Composition holds for CQT expressions
Attribution composition computes the attribution of a composed CQT expression.

Assume without loss of generality the following CQT expressions E_1, E_2, E_3 defined by the formulas f, g, and h respectively s.t.
$E_1 \overset{\text{def}}{=} f = (p_1 \wedge p_2 \wedge \dots \wedge p_n \wedge q)$ where q is the only IDB in E_1
$E_2 = q \overset{\text{def}}{=} g = (r_1 \wedge r_2 \wedge \dots \wedge r_m)$ where $r_i \in d$
$E_3 \overset{\text{def}}{=} h = (p_1 \wedge p_2 \wedge \dots \wedge p_n \wedge r_1 \wedge r_2 \wedge \dots \wedge r_m)$
Again, we assume that variables in formula g of E_2 are renamed and reordered appropriately. The p's and r's may now include theta comparisons in addition to relational predicates with constants. We further assume, for convenience, that obvious redundancies are reduced (e.g. $(X < 10) \wedge (X < 5)$ reduces to simply $(X < 5)$)

Given E_1 defined on $d' = d \cup \{q\}$ and r, the result of evaluating E_1 on d', attribution composition computes the [comprehensive | source | relevant] attribution of result r in terms of d as defined by attr(r, E_3, d).

Lemma 4.9 $(a_3,h) \in A_3$ is a comprehensive attribution for E_3 if and only if $(a_3,h) \in$ Compose (A_1,f).

This case is no different than for CQ expressions. That variables may now also appear in arithmetic comparisons does not affect their substitutions which are bound by the relational predicates.

Lemma 4.10 $(a_3,h) \in A_3$ is a source attribution for E_3 if and only if $(a_3,h) \in$ Compose (A_1,f).

A_1 is the source attribution for E_1. Again, this follows the parallel for CQ expressions. Source attribution is defined by the relational predicates in which the free variables appear.

Lemma 4.11 $(a_3,h) \in A_3$ is a relevant attribution for E_3 if and only if $(a_3,h) \in$ Compose (A_1,f).

A_1 is a relevant attribution for E_1. As with CQ expressions, we need to verify that $relevant(E_3) = relevant(E_1) + relevant(E_2)'$ where $relevant(E_2)' \subseteq relevant(E_2)$ and $relevant(E)$ refers to the relevant variables in E and likewise for $free(E)$; $bound(E)$. In other words, we want to verify that the relevant variables in E_3 are made up of the relevant variables in E_1 and the relevant variables in E_2. Because E_3 is the unification of E_1 and E_2, however, we avoid the problem observed in strict equivalence of identifying interactions between relational predicates and arithmetic comparisons. We form *relevant*(E_2)*'* as we formed A_2' previously. We attribute only the relevant variables in q on the expression E_2.

We note that arithmetic comparisons may now constrain relational predicates containing free variables or relational predicates joined to predicates containing free variables. In addition, arithmetic comparisons may join relational predicates. However, comparisons in the r_j's of E_3 appear in E_2 and comparisons in the p_i's of E_3 appear in E_1. Furthermore, a comparison in the r_j's cannot include variables from the p_i's and vice versa, unless those variables appear in the IDB q of E_1. With these observations in mind, we proceed as in the case for CQ expressions.

Case(→)
Pick some relevant attribution $(a_3,h) \in A_3$ and split it: Project a_3 onto f and g.
We need to establish that $a_3(f) = (a_1,f) \in A_1$ and $a_3(g) = (a_2,g) \in A_2' \subseteq A_2$ where A_2' as the attribution for the tuple defined by $a_1 \cap a_2$, a tuple in q.

A substitution c/X is in a substitution list for a_3 because either X is free in E_3 or c/X joins two relational predicates, at least one of which is recursively joined to a relational predicate containing a free variable or is a constant from the original query expression that appears in a relational predicate recursively joined to a predicate containing a free variable of E_3.

Case 1. $X \in relevant(E_3)$ and $X \in free(E_3)$. $Free(E_3) = X \in free(E_1) \subseteq relevant(E_1)$ by definition of the equivalence of E_1 and E_3. For $X \in free(E_3) = Y \in free(E_2)$, Y must also be free in E_1 because E_2 is q in E_1 (e.g. $Y \in free(E_2) \rightarrow Y \in free(E_1)$). Consequently, at least for the relevant variables in E_2 that are free, we know $a_2 \in A_2' \subseteq A_2$

Case 2. $X \in relevant(E_3)$ joins relational predicates to a recursively joined set of relational predicates or X constrains one predicate in a recursively joined set of relational predicates (e.g. X is a constant in the formula or in an arithmetic comparison). All such predicates are in the set p_i and at least one joined predicate contains a free variable in h. Then X is relevant in E_1 so $a_3(f) = a_1$ for $(a_1,f) \in A_1$.

Case 3. $X \in relevant(E_3)$ is like Case 2 except all such predicates are in the set r_j. Then X is relevant in E_2 so $a_3(g) = (a_2,g) \in A_2' \subseteq A_2$ where A_2' as the attribution for the tuple defined by $a_1 \cap a_2$, a tuple in q. (recall that $g \overset{\text{def}}{=} q$ in E_1).

Case 4. $X \in relevant(E_3)$ appears in both some predicate p_i and some predicate r_j. Then X must appear in predicate q of E_1 (X is not free in E_1 so it must be bound in E_1 and appear in q in order to appear in both p_i and r_j in E_3). Therefore X is relevant in E_1 so $a_3(f) = a_1$ for $(a_1,f) \in A_1$.

It is possible that $a_3(g)$ is empty, which occurs when we consider a Cartesian Product and then do not restrict variables between arguments to the Cartesian Product. In this case, attribution relevance is trivially true in the p_i's.

From here, we do the same unfolding as before and conclude that **Compose** returns $a_1 \circ a_2(h)$ which is a_3.

Case (←)
Pick some random substitution list $a_1 \circ a_2$ as before and verify that $(a_1 \circ a_2, h) \in A_3$.
Proof by contradiction.
Suppose not. Then there must be a substitution $c/X \in a_1 \circ a_2$, $c/X \notin a_3$, or $c/Y \in a_3$, $c/Y \notin a_1 \circ a_2$.

Case 1. Pick Y. If Y is relevant in E_3, then Y must constrain the free variables in h in some way.
If Y is a free variable, then $c\ /Y \in a_1$, a contradiction.
If Y constrains a predicate containing a free variable through some recursively joined set of predicates amongst the p_i's, then $c/Y \in a_1$, a contradiction.
If Y constrains a predicate containing a free variable through some recursively joined set of predicates amongst the r_j's and is $a_1 \circ a_2$ as assumed above, then $c/Y \in a_2$, a contradiction.
If Y is relevant and appears in both some predicate p and some predicate r then $c/Y \in a_1$, a contradiction (see Case 4 for the (→) direction).
Therefore, by contradiction, we conclude that there is no $c/Y \in a_3$, $c/Y \notin a_1 \circ a_2$.

Case 2. Pick X.
If $c/X \in a_1$ because it is a free variable in E_1, then by definition, $c/X \in a_3$, a contradiction.
If $c/X \in a_1$ because it constrains a free variable through predicates p_i, then $c/X \in a_3$, a contradiction.
If $c/X \in a_1$ and appears in both in q's and p's, then $c/X \in a_3$, a contradiction.
Now we need to be careful. Remember that a_2 is selected from attributing $a_1(g)$. It is possible for $a_1(g) = \{\}$ as in the case of Cartesian Product. attr($\{a_1(g)\}$, E_2, d') is non-empty only when there is a relevant variable in q.
If $c/X \in a_2$ because it is free in E_2 and free in E_1, then we know $c/X \in a_3$, a contradiction.
If $c/X \in a_2$ because it is free in E_2 and bound and relevant in E_1, then we know $c/X \in a_3$, a contradiction.
If $c/X \in a_2$ because it is bound in E_2, occurs among the predicates r_j and constraints a free variable in E_2 that is relevant in E_1 (through predicate q in E_1), then we know $c/X \in a_3$, a contradiction. □

Therefore, we conclude that attribution composition computes the attribution of a composed CQT expression. □

4.5 Adding explicit equality

Adding explicit equality to CQT expressions challenges our intuitions about the attribution of equivalent queries, but not necessarily in unexpected ways. The source of a variable is determined syntactically by occurrences of that variable in the expression. Logically, we say that the source of a variable is the predicate by which we limit (for purposes of safety) the values of a particular domain. Example 3.12 in Section 3 contrasted natural joins and explicit equality. We present a more abstract example here.

Example 4.25 Explicit equality

$E_{20} = \{XY \mid p(XYZ) \wedge q(UVW) \quad (X = U)\}$
$E_{21} = \{XY \mid p(XYZ) \wedge q(XVW)\}$
$E_{22} = \{XZ \mid p(UWWX) \wedge (X = U)\}$

In E_{21}, both predicates p and q may be said to limit values of X for purposes of safety. In E_{20}, predicate q does limit values of X, but only indirectly through an explicit comparison to U. In E_{22}, note that all variables are limited in the same predicate. More particularly, from the perspective of equivalence the examples introduce a slight irregularity into the containment map. We either implicitly push all equalities into the predicates (for example, eliminating variable U as in E_{21}) or rename all variables so that no variable name appears more than once as in E_{20}; all equalities are than explicit. Without the change, the containment map takes X and U in E_{20} to X in E_{21}. Mapping from E_{21} to E_{20}, however is less clear. To what variable in E_{20} do we map E_{21} □

Rather than resolving the problem of explicit equality by either pushing equalities into relational predicates or renaming all variables, we suggest that the syntactic difference may prove useful for purposes of attribution. Under this interpretation, different relations that include the same domain may use the same domain variable to indicate multiple sources for that domain. In this way, we use the introduction of explicit equality to help differentiate attribution.

Example 4.26 Source attribution and explicit equality

E_{23} = {HNAME | regions(HNAME, REGION) ∧ sites("Imperial Palace", REGION) }
E_{24} = {HNAME | hostels(HNAME, PRICE, STATION) ∧ sites("Nakamise Dorsi", REGION) ∧
(REGION = STATION)}

E_{23}, adapted from Q2 in Section 3, attempts to locate hotels by the "Imperial Palace". It does so by matching the REGION in which the Imperial Palace is located, to the REGION in which individual hotels are located. Here, the two relations draw from the same domain so both relational predicates are considered sources for values of REGION.

E_{24}, adapted from Q5 in Section 3, attempts to locate hostels by "Nakamise Dorsi". However, the relation for hostels does not know about the domain of REGIONs. Rather, the query uses the knowledge that many train stations are named for the region in which they reside. As a consequence, we find hostels by equating values from the REGION domain with values from the STATION domain. □

Associating attribution with the syntax of a calculus expression allows us to distinguish between the concept of the natural join and the theta join (Ullman 1988). For purposes of attribution, in the natural join, two relations implicitly serve as sources for the same attribute domain. In theta-join, two attribute values, possibly from dissimilar domains, are explicitly compared.

Using the syntax of explicit equality to distinguish between different sources, however, clearly compromises the equivalence of comprehensive, source, and relevant attributions of strictly equivalent queries. We therefore offer

Observation 4.10 Attribution of strictly equivalent CQT$^+$ expressions
Though E_1 and E_2 are equivalent CQT$^+$ expressions (CQT expressions with explicit equality), then their [comprehensive | source | relevant] attributions, A_1 and A_2, are **not** necessarily equivalent. To see this, we need only recognize that the source attributions of equivalent expressions no longer necessarily map to one another as in E_{23} and E_{24} above. Comprehensive and relevant attribution suffer from the same issue. Although predicates map, there is not necessarily a consistent way of mapping domain variables between equivalent expressions. □

4.6 Adding union

In this next extension, we consider the addition of union into the query language. Unlike earlier extensions, union allows us to introduce and eliminate predicates from equivalent expressions. As a result, we first redefine our concept of attribution to account for union. We conclude that for the different types of attribution, the attribution of strictly equivalent queries are no longer necessarily equivalent. With some minor adjustments to the algorithm, however, we can show that attribution does continue to compose.

4.6.1 Attribution concept

Much as with the introduction of θ-comparison, the DRC imposes safety constraints on our introduction of disjunction in the language to support the semantics of algebraic union. In particular, the disjunction of two predicates must have the same set of arguments much as the algebraic condition on union requires union compatibility (Ullman 1988). As a further simplification for defining attribution in the presence of union, we assume prenex, disjunctive normal form (DNF) as the canonical form for all CQTU expressions. We know that we can transform a safe calculus expression into this form. A CQTU query therefore has the form:

$$\{X_1,\ldots,X_n \mid f_1(X_1,\ldots,X_n) \vee\ldots\vee f_m(X_1,\ldots X_n)\}$$

Every disjunct f_j is a CQT query that alone may make the expression true. In light of disjunction, we therefore generalize our original intuition for attribution. Attribute each disjunct as an independent CQT$^+$ query.

Definition 4.15 Attribution of the union of CQT^+ expressions
The [comprehensive | source | relevant] attribution of the disjunction of CQT^+ expressions is the union of the corresponding attributions for each constituent disjunct. □

Example 4.27 Attribution of the union of CQT^+ expressions

$E_{25} = \{A \mid p(ABC) \vee q(ABC)\}$

The [comprehensive | source | relevant] attribution for the expression is therefore the attribution of {A | p(ABC)} combined with the attribution of {A | q(ABC)} □

We actually saw several examples of unions from the examples in Section 3 beginning with Example 3.6. In the case of the union of CQ expressions, we know that there is a unique minimal equivalent (Ullman 1989). We find the unique minimal equivalent by minimizing each disjunct independently and then removing disjuncts that are contained by other disjuncts in the same expression. Under these limited circumstances, then, we can certainly argue that, for the unique minimal expression, the comprehensive, source, and relevant attributions are the same. For the general case of attribution equivalence of strictly equivalent queries, however, attribution equivalence breaks down with the introduction of union.

4.6.2 Multiple derivations – strict equivalence

For attribution, which is based upon substitutions, the problem posed by the introduction of union is immediately clear. Disjunction allows the introduction of new predicates, hence new variables and new substitutions. The containment condition for equivalent queries and the attendant mapping between attributions for equivalent expressions therefore breaks down. Example 3.7 of Section 3 offered one example of how attribution breaks down under union. Here, we consider a more abstract case. Consider the following equivalent expressions.

Example 4.28 Attribution of strictly equivalent expressions with disjunction

$E_{26} = \{A \mid p(ABC) \vee (p(ABC) \wedge q(ABC))\}$
$E_{27} = \{A \mid p(ABC) \vee (p(ABC) \wedge (C < 10))\}$
$E_{28} = \{A \mid p(ABC)\}$

The three queries are equivalent because the second disjunct in E_{26} and E_{27} is contained by the first disjunct. E_{26} and E_{27} therefore reduce to E_{26}. However, the comprehensive attribution for the first expression includes substitutions in the predicate q which do not map to the other equivalent expressions. Perhaps more obvious, we may regard q as a source for the attribute values of A in E_{26} although neither of the equivalent expressions reference q. For relevant attribution, we see that a variable, relevant in one disjunct, can prove irrelevant in a disjunct of the same expression or to an equivalent expression. In E_{27}, the attribute variable C is relevant in the second disjunct but neither in the first disjunct of the same expression nor in the third expression. □

That attribution breaks down under union corresponds to our intuitions about attribution. Attribution can provide corroborating information about the quality of a particular query result or the values in a particular result. Though redundant, attribution may also provide references to non-redundant ancillary information. Finally, from an intellectual property perspective, whether a source proves redundant or not, proper acknowledgement and perhaps remuneration is only appropriate.

4.6.3 Multiple derivations - composition

While attribution equivalence breaks down for strictly equivalent queries, we see that attribution continues to compose. As observed earlier for the relevant attribution of CQT expressions, composition assumes that we begin with a single formula and unfold the IDB. Composition does not reduce redundant disjuncts. We reason that we may unfold redundant disjuncts as easily as any other disjunct in the disjunction of CQT^{+} expressions (assuming also the appropriate renaming and reordering to avoid conflict in multiple occurrences of the same predicate or domain variable in the same disjunct).

We first update our algorithm to account for disjunctions. Then, we prove that the algorithm holds for the introduction of safe disjunction assuming that queries are expressed in canonical form.

To update the algorithm, we must first recall that the attribution of the expression is now the union of the attributions of each disjunct. We assume that the definition of any IDB may also include disjunction but that all IDB definitions are expressed in canonical form as well (i.e. the disjunction of conjuncts). The accumulation of disjuncts must therefore distribute in the original expression.

Example 4.29 Attribution of a composed expression with nested disjunction

$$E_{29} = \{A \mid p(ABD) \vee q(ACE)\}$$
$$E_{30} \overset{def}{=} q(ACE) = \{ACE \mid (r(ABC) \wedge s(CDE)) \vee t(ACE)\}$$
$$E_{31} = \{A \mid p(ABD) \vee (r(ABC) \wedge s(CDE)) \vee t(ACE))$$

E_{21} is an expression with an IDB in the second disjunct. The IDB, which we label E_{30}, itself contains a disjunction. Unifying the IDB gives E_{31}. Note the necessary variable renaming. Attribution is defined in terms of the base relations. As before, we want to discover whether we may iteratively attribute E_{29} and E_{30} in lieu of unifying the expression a priori.

Algorithm 4.2Attribution composition for $CQT^{+}U$ expressions

Compose (A, s) where A is the attribution for s, a disjunction of CQT^{+} sub-formulas, each of which may itself be a disjunction of CQT^{+} sub-formulas.

```
Compose (A, s) {                                                        (a)
if $s = \varnothing$ then return { }                                    (b)
else pick $f_i$ a disjunct in $s = f_1 \vee f_2 \vee \dots f_x$         (c)
    $s := f_1 \vee f_2 \vee \dots f_{i-1} \vee f_{i+1} \vee \dots \vee f_x$   (d)
    $A' := \{(a,f) \mid (a,f) \in A \text{ and } f = f_i\}$             (e)
    Compose (A, s) $\cup$ ComposeD ($A',f_l$)}                          (f)

ComposeD ($A,f$) {                                                      (1)
if $f$ has no $q$'s then return $A$                                     (2)
else pick $q_i$, an IDB in $f$                                          (3)
    $f := p_1 \wedge p_2 \wedge \ldots \wedge q_{i-1} \wedge q_{i+1} \ldots q_m$                           (4)
    ComposeD (Unfold ($A, q_i$),$f$) }                                  (5)

Unfold ($A, q$) {                                                       (6)
if $A$ is $\varnothing$ then return { }                                 (7)
else pick $(a,f) \in A$                                                 (8)
    let $g$ be the formula for IDB $E$ representing $q$                 (9)
    let $u$ be the unifier for $h = unify(f,g)$                         (10)
    let $E'$ be $E$ as defined by $g$ with the renaming of $u$          (11)
    $B$ = attr( $E'( a(q)/x )$, $d'$)                                   (12)
    Rewrite ($B$, $u(a - a(q))$, $h$) $\cup$ Unfold ($A - \{(a,f)\}$, $q$) }   (13)

Rewrite ($B$, $a$, $h$) {                                               (14)
if $B$ is $\varnothing$ then return { }                                 (15)
else pick $(b,g) \in B$                                                 (16)
    $\{<\{a \circ b\},h>\}$ $\cup$ Rewrite ($B - \{(b,g)\}$, $a$, $h$)  (17)     □
```

This is the same algorithm as that presented for CQ expressions with the exception being lines (a) – (f). What was formerly called "Compose" we renamed "Compose Disjunct" or "ComposeD." As declared in line (a), "Compose" is now a function that recurses down the disjuncts in the formula for the query expression. We call "ComposeD" on each disjunct as if it were an isolated CQT$^+$ query. The attribution of the expression is then the union of the attributions from calling "ComposeD" on each disjunct. Because each substituion is defined for only one disjunct in the query expression, line (e) ensures that we ComposeD on each disjunct with only those substitutions applicable to a respective disjunct. We then propose:

Theorem 4.5 Attribution composition

Our algorithm for attribution composition computes the attribution for the union of composed CQT$^+$ expressions. Assume the following CQT+ expressions E_1, E_2, E_3 defined by the formulas f, g, and h respectively as:

$E_1 \overset{def}{=} f = (p_1 \wedge p_2 \wedge \ldots \wedge p_n \wedge q) \vee (t_1 \wedge t_2 \wedge \ldots\ )$ where q is the only IDB in E_1

$E_2 = q \overset{def}{=} g = (r_1 \wedge r_2 \wedge \ldots \wedge r_m) \vee (s_1 \wedge s_2 \wedge \ldots \wedge s_o)$ where $r_i, s_i \in d$

$E_3 \overset{def}{=} h = (p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m) \vee (p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o) \vee (t_1 \wedge \ldots\ )$

Furthermore, we know that ($r_1 \wedge r_2 \wedge \ldots \wedge r_m$) and ($s_1 \wedge s_2 \wedge \ldots \wedge s_o$) are union compatible with schema defined by the IDB q.

Given E_1 defined on $d' = d \cup \{q\}$ and r, the result of evaluating E_1 on d', attribution composition computes the [comprehensive | source | relevant] attribution of result r in terms of d as defined by attr(r, E_3, d).

Lemma 4.12 $(a_{3,}h_i) \in A_3$ is a comprehensive attribution for E_3 if and only if $(a_{3,}h_i) \in$ Compose $(A_{1,}f)$.

Case (→)
Pick a random substitution $(a_{3,}h_i) \in A_3$. Consider the following possibilities:

$h_i = (p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$
$h_i = (p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o)$
$h_i = (t_1 \wedge \ldots)$

If $h_i = (t_1 \wedge \ldots)$ then we know that $(a_{3,}h_i) = (a_{1,}h_i) \in A_1$ because h_i is a disjunct in the formula for E_1 (see Algorithm 4.2 line (c)). For A' on $f_i = x_1 \ldots x_n = h_i$ we know that $I(h_i(a_3/x)) = true$ so $I(f_i(a_3/x)) = true$. There are no IDB in f_i so $(a_{3,}h_i) \in$ **ComposeD**$(A_{1,}f_i) \in$ **Compose**(A_1, f).

If $h_i = (p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$ in A_3 then we can say that $a_3(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$, $(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m) = a_1$, $(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m) \in A_1$ because $\forall i$, $I(p_i(a_3(p_i)/x)) = true$ and $\forall j$, $I(r_j(a_3(r_j)/x)) = true$ and q is defined by the formula g. Or, to be more precise, $r_1 \wedge \ldots \wedge r_m$ is a disjunct of g that makes g true. Similarly, we know that $(a_3(r_1 \ldots r_m), g) = (a_2(r_1 \ldots r_m) \in A_2' \subseteq A_2$ (where A_2' is the attribution for tuple $a_1 \cap a_2$, a tuple in q). **Compose** calls **ComposeD** on $f_i = (p_1 \ldots q)$ with $A' = (a_1, f_i)$ in line (f) of Algorithm 4.2. **ComposeD** passes A' to **Unfold**. **Unfold** calls attr($\{a_3(q)\}, E_2, d'$) which we already know is $A_2' \subseteq A_2$. **Unfold** is applied to every value of A' so certainly it calls itself on a_1 which we have already seen makes E_1 true. **Unfold** calls **Rewrite** with a_1 and A_2' so certainly it is applied to a_2. But **Rewrite** is called on h_i, the unification of $p_1 \ldots q$ and g and returns $(a_1 \circ a_2)$ which is a_3.

If $h_i = (p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o)$ then we apply the same analysis as before, knowing that $h_i = (s_1 \wedge \ldots \wedge s_o)$ is a disjunct of g that also makes q true. As a consequence, it produces $A_2'' \subseteq A_2$ from attr($\{a_3(q)\}, E_2, d'$) and we arrive at the same conclusion as before.

Case (←)
If (unify(f,g)) results in the disjuncts:

$(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$,
$(p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o)$ or
$(t_1 \wedge \ldots)$

does every $(a_1, t_1 \wedge \ldots)$ or $(a_1 \circ a_2, p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$ or $(a_1 \circ a_2, p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o)$ appear as a substitution in A_3? Pick some arbitrary a_1 from a pair in A_1. If you picked some (a_1, $t_1 \wedge ...$) then we know that a_1 makes disjunct ($t_1 \wedge ...$) true. But because t is also a disjunct of E_3, if a_1 makes the disjunct true, then certainly it makes E_3 true therefore (a_1, $t_1 \wedge ...$) $\in A_3$.

Now if the pair $a_1 \in A_1$ is for disjunct ($p_1 \wedge p_2 \wedge ... \wedge p_n \wedge q$) we want to pick an a_2 but not an arbitrary a_2. From Algorithm 4.2, **Compose** creates A_2' from attr($\{a_1(g)\}$, E_2, d'). So pick any a_2 from a pair $\in A_2'$. We know $a_1 \circ a_2$ paired with ($p_1 \wedge ... \wedge p_n \wedge r_1 \wedge ... \wedge r_m$) $\vee$ ($p_1 \wedge ... \wedge p_n \wedge s_1 \wedge ... \wedge s_o$) appears in A_3 if it makes E_3 true. $\forall i$, $I(p_i(a_3(p_i)/x)) = true$ and $\forall j$, either $I(r_j(a_3(r_j)/x)) = true$ or $I(s_j(a_3(s_j)/x)) = true$. But is either disjunct true at the same time that the p's are true? Because we know that a_2 is from a pair in A_2' by construction, we know that a_2 makes E_2 true. Therefore, we know that ($a_1 \circ a_2$, ($p_1 \wedge ... \wedge p_n \wedge r_1 \wedge ... \wedge r_m$) $\vee$ ($p_1 \wedge ... \wedge p_n \wedge s_1 \wedge ... \wedge s_o$)) $\in A_3$. □

Lemma 4.13 (a_3,h_i) $\in A_3$ is a source attribution for E_3 if and only if (a_3,h_i) $\in$ Compose (A_1,f) *w*here A_1 is the source attribution for E_1.

Case (→)
Pick a random substitution (a_3,h_i) $\in A_3$. Consider the following possibilities:

$h_i = (p_1 \wedge ... \wedge p_n \wedge r_1 \wedge ... \wedge r_m)$
$h_i = (p_1 \wedge ... \wedge p_n \wedge s_1 \wedge ... \wedge s_o)$
$h_i = (t_1 \wedge ...)$

Regardless of which alternative is chosen, the source attribution consists of the free variables (and the accompanying variables in the associated relational predicate(s)).

If $h_i = (t_1 \wedge ...)$ then we know then we know that $a_3(t_1 \wedge ...)$, ($t_1 \wedge ...$) $\in A_1$ because a_3 identifies the disjunct ($t_1 \wedge ...$) of E_3. But ($t_1 \wedge ...$) is also a disjunct of E_1, so this holds trivially. Note that there is no IDB in ($t_1 \wedge ...$) so Algorithm 4.2 line (2) returns the original source attribution for the disjunct ($t_1 \wedge ...$) for **Compose**(A_1, ($t_1 \wedge ...$)).

If $h_i = (p_1 \wedge ... \wedge p_n \wedge r_1 \wedge ... \wedge r_m)$ then we observe that $a_3(p_1 \wedge ... \wedge p_n \wedge q)$, ($p_1 \wedge ... \wedge p_n \wedge q$) is a pair $\in A_1$ because for predicates p_i, $E_1 \subseteq E_3$ and for predicate $q \in E_1$, q is defined in terms of the free variables of E_2 which is unfolded in a_3. Similarly, we can say that (a_3($r_1 \wedge ... \wedge r_m$)), q = (a_2,g) $\in A_2' \subseteq A_2$ where A_2' is the source attribution of $a_1 \cap a_2$, a tuple of q. **Compose** passes $f' = (p_1 \wedge ... \wedge p_n \wedge q)$ to **ComposeD** with source attribution A' defined in terms of the p's and q's. **ComposeD** passes A' to **Unfold** with $q = (r_1 \wedge r_2 \wedge ... \wedge r_m) \vee (s_1 \wedge s_2 \wedge ... \wedge s_o)$. **Unfold** is called on every source substitution in A' so certainly it is called on a_1. **Unfold** calls for attr($\{a_1(g)\}$, E_2, d') which we know includes the source substitutions $A_2' \subseteq A_2$ where the formula in the attribution pair is the disjunct ($r_1 \wedge r_2 \wedge ... \wedge r_m$). **Rewrite** is called on every element of A_2' so eventually it is called on a_2. But **Rewrite** pairs $a_1 \circ a_2$ with

h_i a disjunct $(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$ of unify(f,g) in line (10) of Algorithm 4.2. This, then, is just a_3. The same reasoning applies for the disjunct $(s_1 \wedge s_2 \wedge \ldots \wedge s_o)$ from the attribution in **Unfold**.

Case (←)
If (unify(f,g)) results in the disjuncts:

$(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$,
$(p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o)$ or
$(t_1 \wedge \ldots)$

does every $(a_1, t_1 \wedge \ldots)$ or $(a_1 \circ a_2, p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$ or $(a_1 \circ a_2, p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o)$ appear as a substitution in A_3? For pairs $(a_1, t_1 \wedge \ldots)$ then we know that a_1 is a source substitution $(t_1 \wedge \ldots)$. But because t is also a disjunct of E_3, if a_1 is a valid source substitution for E_1, then certainly it is likewise for E_3 therefore $(a_1, t_1 \wedge \ldots) \in A_3$.

For pairs involving an $a_1 \circ a_2$ pick some arbitrary a_1 from a pair in A_1. Now pick an a_2 from $A_2' \subseteq A_2$ generated by the attr$(\{a_1(g), E_2, d'\})$ in **Unfold**. This will give a substitution a_2 either in $(r_1 \wedge r_2 \wedge \ldots \wedge r_m)$ or $(s_1 \wedge s_2 \wedge \ldots \wedge s_o)$. We know that $a_1 \circ a_2$ paired with $(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$ or $(p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o)$ makes A_3 true if it makes E_3 true. And we know a_1 makes the p's true just as a_2 makes the r's or the s's true by construction. Therefore, we know $(a_1 \circ a_2, (p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)) \in A_3$ and $(a_1 \circ a_2, (p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o)) \in A_3$ □

Lemma 4.14 $(a_3,h) \in A_3$ is a relevant attribution for E_3 if and only if $(a_3,h) \in$ Compose (A_1,f).
Where A_1 is a relevant attribution for E_1. As in prior cases, we need to verify that $relevant(E_3) = relevant(E_1) + relevant(E_2)'$ where $relevant(E_2)' \subseteq relevant(E_2)$ and $relevant(E)$ refers to the relevant variables in E and likewise for $free(E); bound(E)$. We form $relevant(E_2)'$ as we formed A_2' previously. We attribute only the relevant variables in q on the expression E_2. With disjunction, there is the additional complexity of tracking relevance in each disjunct.

Case (→)
Pick a random substitution $(a_3, h_i) \in A_3$. Consider the following possibilities:

$h_i = (p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$
$h_i = (p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o)$
$h_i = (t_1 \wedge \ldots)$

Suppose $h_i = (t_1 \wedge \ldots)$. We also know that $(t_1 \wedge \ldots)$ is a disjunct of E_1 which means that $(a_3, (t_1 \wedge \ldots)) \in A_1' \subseteq A_1$. Because there are no IDB in this disjunct, we know that the call to

ComposeD on $(t_1 \wedge \ldots)$ with $A_1' \subseteq A$ for pairs $(a_3, (t_1 \wedge \ldots))$ simply returns A_1'. So we conclude $(a_3, (t_1 \wedge \ldots)) \in$ **Compose**$(A_1, (t_1 \wedge \ldots))$.

If $h_i = (p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$ then we consider the same cases as for relevance in CQ. However, we must now consider the cases for each disjunct.

Case 1. $X \in relevant(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$ and $X \in free(\, p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m\,)$. We know that $free(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m) = X \in free(E_1) \subseteq relevant(E_1)$ by definition of the equivalence of $p_1 \wedge \ldots \wedge p_n \wedge q$ and $h_i = p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m$. Consequently, at least for relevant variables in the disjunct $(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$ that are free, we know $a_2 \in A_2' \subseteq A_2$.

Case 2. $X \in relevant(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$ joins relational predicates to a recursively joined set of relational predicates or X constrains one predicate in a recursively joined set of relational predicates (e.g. X is a constant or X appears multiple times in a single relation). All such predicates are in the set p_i and at least one joined predicate contains a free variable in $h_i = p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m$. Then X is relevant in E_1 so $a_3(f) = a_1$ for $(a_1, p_1 \wedge \ldots \wedge p_n \wedge q) \in A_1$.

Case 3. $X \in relevant(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$ as in Case 3 of Lemma 4.6 where the X's appear only in the r_j's. Then $X \in relevant(r_1 \wedge \ldots \wedge r_m)$ which implies $X \in relevant(E_2)$ so $a_3(g) = (a_2,g) \in A_2' \subseteq A_2$ where A_2' as the attribution for the tuple defined by $a_1 \cap a_2$, a tuple in q. Of course A_2' may also include some substitutions from other disjuncts in the definition of q (e.g. $s_1 \wedge \ldots \wedge s_o$).

Case 4. $X \in relevant(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$ appears in both some predicate p_i and some predicate r_j. Then X must appear in predicate q of E_1 (X is not free in E_1 so it must be bound in E_1 and appear in q in order to appear in both p_i and r_j in E_3). Therefore X is relevant in E_1 so $a_3(f) = a_1$ for $(a_1, p_1 \wedge \ldots \wedge p_n \wedge q) \in A_1$.

It is possible that $a_3(g)$ is empty, which occurs when we consider a Cartesian Product and then do not restrict variables between arguments to the Cartesian Product (i.e. no θ comparisons). In this case, attribution relevance is trivially true in the p_i's.

From here, we do the same unfolding as before and conclude that **Compose** returns $(a_1 \circ a_2, (p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)) \in A_3$. We can do the same analysis for $h_i = p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o$ or any other disjunct of q.

Case (←)
Pick some random substitution from **Compose**: $(a_1, (t_1 \wedge \ldots))$ or $(a_1 \circ a_2, f')$ where f' is a disjunction $(p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m)$ or $(p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o)$ and verify that it appears in A_3.

Proof by contradiction.
Suppose not. Then there must be a substitution:

$c/X \in a_1$ where $(a_1,\ (t_1 \wedge \ldots\ )) \notin A_3$ or some

$c/X \in a_1 \circ a_2$ where $(a_1 \circ a_2, p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m\ ) \notin A_3$ or some

$c/X \in a_1 \circ a_2$ where $(a_1 \circ a_2, p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o\ ) \notin A_3$.

But we know $(t_1 \wedge \ldots\ )$ is a disjunct in E_3 so if a_1 is relevant in the t's for E_1 then it must still be relevant in the same disjunct of E_3. A contradiction.

If $c/X \in a_1$ because it is free in E_1 then by definition, $c/X \in a_3$, a contradiction.
If $c/X \in a_1$ because it constrains a free variable through the p's then $c/X \in a_3$.
If c/X appears in both the p's and predicate q, then $c/X \in a_3$ by definition.
Now we need to be careful. Remember that a_2 is selected from attributing $a_1(g)$. It is possible for $a_1(g) = \{\}$ as in the case of Cartesian Product. attr($\{a_1(g)\}$, E_2, d') is non-empty only when there is a relevant variable in q. Recall that E_2 is in DNF so the free variables are the same in each disjunct of E_2.
If $c/X \in a_2$ because it is free in E_2 and free in E_1, then we know $c/X \in a_3$, a contradiction.
If $c/X \in a_2$ because it is free in E_2 and bound and relevant in E_1, then we know $c/X \in a_3$, a contradiction.
If $c/X \in a_2$ because it is bound in E_2, occurs among the predicates r_j of a disjunct in E_2 (similarly for the other disjuncts of E_2 i.e. s_j) and constraints a free variable in E_2 that is relevant in E_1 (through predicate q in E_1), then we know $c/X \in a_3$, a contradiction. □

Therefore, we conclude that attribution composition computes the attribution of a composed expression. □

By representing our expressions in DNF, we can treat each disjunct independently and compose in a depth first manner across all disjuncts and all IDB. As before, we can easily imagine unfolding successive levels of IDB.

4.7 Adding negation

Negation, in general, poses problems for query evaluation (Abiteboul, Hull, and Vianu 1995). Likewise, negation presents problems for attribution. From Section 3, the intuition behind attribution for negation corresponds to the logical interpretation of safe expression. We can confirm the truth of a negated assertion (fact in the database) by verifying that the (positive) assertion itself does not exist in the database. Unfortunately, this intuition breaks down under composition of queries with negation. We identify a subset of queries with negation under which composition is preserved.

4.7.1 Attribution concept

As indicated in Section 3, to verify that the (positive) assertion does not exist, the attribution must therefore consider (include) every true substitution for the negated sub-formula. We first illustrated this intuition in Example 3.13 and Example 3.14 of Section 3.

Example 4.30 Attribution for an expression with negation

$E_{32} = \{ABC \mid r(ABC) \wedge \neg\, s(ABC)\}$

To verify that a substitution <1/A, 2/B, 3/C> is in the attribution for the expression, we must not only verify that $I(r(1/A, 2/B, 3/C)) =$ true but also that for every substitution <x/A, y/B, z/C> such that $I(s($ x/A, y/B, z/C $)) =$ true, $x \neq 1$ or $y \neq 2$ or $z \neq 3$. □

Moreover, if there is more than one negated predicate, we need to confirm that a valid substitution for the expression does not make any of the negated predicates true. We would do so by confirming that a substitution for the formula does not include any true substitution for any negated predicate.

4.7.2 Types of attribution

To formalize attribution in the context of negation, we introduce a few additional assumptions. First, using standard rules, all negations are pushed down to the level of individual predicates. The negation of an arithmetic comparison is simply expressed as its logical converse (e.g. $\neg\, (X < Y) \equiv (X \geq Y)$). Second, formulas continue to be flattened as the disjunction of conjuncts where all conjuncts are either positive or negative predicates or theta comparisons. Within each disjunct, negated predicates are limited for safety as per the syntactic rules described earlier. A formula is therefore a disjunction of conjuncts of the form:

$$p_1 \wedge p_2 \wedge \ldots \wedge p_n \wedge \neg\, q_1 \wedge \neg\, q_2 \wedge \ldots \wedge \neg\, q_m \wedge t_1 \wedge \ldots \wedge t_o$$

where the *p*'s are non-negated predicates, the *q*'s are negated predicates, and the *t*'s are theta comparisons. For safety, for each *j* in *m*, every argument in q_j must also appear in some predicate p_i or bound to a constant. Based upon these extensions to address negation, we can now redefine what we mean by attribution.

Definition 4.16 Comprehensive attribution

The comprehensive attribution for an expression in DNF, possibly with negated predicates, is the union of the comprehensive attributions for each disjunct, *f*. The comprehensive attribution for each disjunct is a set of triples $<a, n, f>$ where *a* is a substitution for which the non-negated predicates p_i and θ-comparisons t_o in disjunct *f* evaluate to true and *n* is itself a set of substitutions $\{<b, m, q_j>\}$. The set *n* ranges over all of the negated predicates q_j and includes every substitution *b* that makes q_j true. Assuming that there is no *b* that agrees in the corresponding substitutions for values of *a* $I(q_j(a)) = false$ we may then concludes $I(\neg\, q_j(a)) = true$. By default, *m* is $\varnothing$. □

In source attribution, the intuition is that we want to know the predicates (and their corresponding substitutions) from which values in the query result are drawn. Therefore, only non-negated predicates are considered as possible sources. Negated predicates because they do not match our intuition as a source for values in the result.

Definition 4.17 Source attribution
The source attribution for an expression in DNF, possibly with negated predicates, is the union of the source attributions for each disjunct, *f*. The source attribution for each disjunct is a set of triples <*a', n, f*> where *a'* is a sublist of substitutions *a* for non-negated predicates of *f* that contain free variables and make *f* true. *n* is ∅. □

For relevant attribution we want to consider variables that in some way affect the result. Because of the safety requirement, renaming any variable in a negated predicate would compromise the expression. As a consequence, any variable in a negated predicate is relevant and we have the same issue as introduced in comprehensive attribution for capturing all appropriate substitutions.

Definition 4.18 Relevant attribution
The comprehensive attribution for an expression in DNF, possibly with negated predicates, is the union of the relevant attributions for each disjunct, *f*. The relevant attribution for each disjunct is a set of triples <*a, n, f*> where *a* is a substitution for all relevant variables in *f* that make *f* true. All variables in the head (free in the formula for the expression) are relevant. In addition, a bound variable is relevant to the result if renaming the variable to some name not already in the expression (or eliminating a constant) would relax a constraint on one or more of the attribute domains in the result relation (free in the formula for the expression). By definition, any variable in a negated predicate is relevant. Therefore, as with comprehensive attribution, *n* is itself a set of substitutions $\{<b, m, q_j>\}$. The set *n* ranges over all of the negated predicates q_j and includes every substitution *b* that makes q_j true. We therefore know that $I(q_j(a)) = false$ and $I(\neg\, q_j(a)) = true$. By default, *m* is ∅. □

4.7.3 Attribution equivalence and composition

Having updated our definition of attribution, we consider the impact of introducing negation on our attribution properties. Determining the equivalence of queries with negation is an open question that has persisted for many years (Abiteboul, Hull, and Vianu 1995). It is not an issue that we will attempt to resolve here. Consequently, claims about the attribution of equivalent queries with negation are also outside the scope of this thesis.

However, as seen in our discussion of attribution for CQT expressions, we can address the issue of attribution composition separately. With the introduction of negation, it is apparent that, in general, the property of composition no longer holds. We cannot calculate the attribution for a query result by recursively tracing backwards through each sub-formula. However, we identify a subset of queries under which composition continues to hold.

First, we notice that, in the general case, nested negations (i.e. $b \equiv \neg(\neg b)$) compromises our ability to compose attribution.

Example 4.31 Intersection of predicates a and b using nested negation
Consider two expressions E_{33} and E_{34} with the following formulas.
$f_{33} = a \wedge \neg (a \wedge \neg b)$
$f_{34} = b \wedge \neg (b \wedge \neg a)$

Logically, we know that $E_{33} \equiv E_{34}$. Indeed when we put E_{33} and E_{34} into canonical form by pushing and distributing the negation, we end up with $f_{33} = f_{34} = a \wedge b$. However, suppose we defined the following IDB:
$c = a \wedge \neg b$
$d = b \wedge \neg a$

A substitution in the attribution of c includes values for variables in a and every substitution that makes b true. Likewise for a substitution in the attribution of d. Consider again our original expressions now defined using IDB c and d.
$f_{33}' = a \wedge \neg c$
$f_{34}' = b \wedge \neg d$

By expanding c and d and pushing down the negations, we know that the source attribution for E_{33} = source attribution for E_{34} = source attribution for $(a \wedge b)$. However, we can equally see that the source attribution for E_{33}' = substitutions in A while the source attribution for E_{34}' = substitutions in b.

Similarly, negations are fully eliminated in the canonical form of E_{33} and E_{34} suggesting that a comprehensive or relevant substitution in the attribution for these expressions will be a single list of variables that make a and b true. However, E_{33}' and E_{34}' contain negated literals suggesting that a substitution will include a list of variables that make a (or b respectively) true and then a set of all substitutions that make c (or d respectively) true. Composition would then recurse on all substitutions in c (or d) rather than a single substitution as in $(a \wedge b)$. □

We can think of the phenomenon in the example above as an additivity property that reflects attribution composition. If R is an expression composed on Q and *r* is a result in both Q and R, then the attribution for *r* in R should at least include the substitutions for the attribution of *r* in Q. Unfortunately, as seen in the example above, composition breaks down when we allow negations to cancel one another.

The problem extends beyond nested negations, however. As demonstrated below, distributing negation over conjunction also violates the additivity observed above.

Example 4.32 Distributing negation over conjunction
Imagine expressions with the following formulas.

$f_{35} = \mathsf{C} \wedge \neg (\mathsf{A} \wedge \mathsf{B})$

$f_{36} = (\mathsf{C} \wedge \neg \mathsf{A}) \vee (\mathsf{C} \wedge \neg \mathsf{B})$

Here, we see that the attribution for the first is not the same as the attribution for the second because of what you associate in the attribution. Logically the two are equivalent. However, a triple in the first expression has $n = \{b, m, (\mathsf{A} \wedge \mathsf{B}) \mid I(b/X(\mathsf{A} \wedge \mathsf{B})) = true\}$. A triple in the second expression looks like either $\{b, m, (\mathsf{A}) \mid I(b/X(\mathsf{A})) = true\}$ or $\{b, m, (\mathsf{B}) \mid I(b/X(\mathsf{B})) = true\}$. It is straightforward to see that for substitutions (a, n ,f) where a is only absent from A or from B but not both, that the substitutions could look quite different. as a consequence, it is clear that negation poses some problems for our intuitions about attribution.
□

However, by further constraining the syntactic rules under which we may negate predicates, we arrive at a rudimentary subset of the DRC where negation is permitted yet attribution composition is preserved.

Definition 4.19 Attributable expression.
To define an attributable expression, we extend the rules for safety presented at the beginning of this Section (Ullman 1988). In particular, we introduce the concept of a negatable formula. Only a negatable formula may be negated and remain attributable.
1. Any atom is a formula and is negatable.
2. The disjunction of non-negated atoms is a negatable sub-formula.
3. The disjunction of negatable sub-formulas is negatable. □

Examples 4.33 Negatable sub-formulas in the safe DRC
$f_{37} = \mathsf{A} \wedge \neg \mathsf{B} \wedge \neg \mathsf{C}$
Where A, B, and C are relational predicates representing base relations. Note that the rules of safety require that every variable appearing in B and C also appear in A.

$f_{38} = \mathsf{A} \wedge (\mathsf{B} \vee \mathsf{C})$
The $(\mathsf{B} \vee \mathsf{C})$ is a negatable sub-formula. When we push the negation into the formula, then the formula becomes the same as the first formula.

$f_{39} = (\mathsf{A} \vee \mathsf{B}) \vee (\mathsf{C} \vee \mathsf{D})$ is a disjunction of negatable sub-formulas that are negatable on their face. However, were either of the expressions already negated, then the formula would no longer be negatable. □

We suggest that the attribution of attributable expressions composes. Because we have updated our definitions of attribution to account for negation, our algorithm for composing attributions requires corresponding updates. We first amend our algorithm for calculating attribution and then prove that, for negatable expressions, that the algorithm calculates the attribution for an extended expression.

Algorithm 4.3Attribution composition for negatable query expressions

Compose (A, s) where A is the attribution for *s*, a disjunction of CQT$^+$ sub-formulas with negated predicates, each of which may itself be a disjunction of CQT$^+$ sub-formulas with negated predicates.

```
Compose (A, s) {                                                    (a)
if $s = \varnothing$ then return { }                                (b)
else pick $f_i$ a disjunct in $s = f_1 \vee f_2 \vee \ldots f_x$    (c)
    $s := f_1 \vee f_2 \vee \ldots f_{i-1} \vee f_{i+1} \vee \ldots \vee f_x$    (d)
    $A' := \{(a,f) \mid (a,f) \in A \text{ and } f = f_i\}$          (e)
    Compose (A, s) $\cup$ ComposeD ($A', f_1$)}                     (f)

ComposeD ($A, f$) {                                                 (1)
if $f$ has no $q$'s then return $A$                                 (2)
else pick $q_i$, an IDB in $f$                                      (3)
    $f := p_1 \wedge p_2 \wedge \ldots \wedge q_{i-1} \wedge q_{i+1} \ldots q_m$    (4)
    if $q_i$ is negated                                             (5)
    then ComposeD (UnfoldN ($A, q_i$), $f$)                         (6)
    else ComposeD (Unfold ($A, q_i$), $f$) }                        (7)

UnfoldN ($A, q_i$) {                                                (8)
if $A$ is $\varnothing$ then return { }                             (9)
else pick some triple $<a, n, f> \in A$                             (10)
    let $g$ be the formula for the definition of $q_i$              (11)
    let $u$ be the unifier for $h = unify(f,g)$                     (12)
    $n' :=$ RewriteN ($n, u, q_i$)                                  (13)
    $\{<u(a), n', h>\} \cup$ Unfold ($A - <a, n, f>, q_i$)}         (14)

RewriteN ($n, u, q_i$) {                                            (15)
foreach triple $<b, \varnothing, q>$ in $n$ where $q = q_i$         (16)
    $n := n - \{b, \varnothing, q\}$                                (17)
let $g$ be the formula for the definition of $q_i$                  (18)
$B =$ attr($u(g), d'$)                                              (19)
$n := n \cup B$}                                                    (20)

Unfold ($A, q$) {                                                   (21)
if $A$ is $\varnothing$ then return { }                             (22)
else pick $(a, n, f) \in A$                                         (23)
```

```
    let $g$ be the formula for IDB $E$ representing $q$                  (24)
    let $u$ be the unifier for $h = unify(f,g)$                          (25)
    let $E'$ be $E$ as defined by $g$ with the renaming of $u$           (26)
    $B = \text{attr}( E'( a(q)/x ), d')$                                 (27)
    Rewrite ($B$, $u(a - a(q))$, $h$) $\cup$ Unfold ($A - \{(a,f)\}$, $q$) }   (28)

Rewrite ($B$, $a$, $h$) {                                                (29)
if $B$ is $\emptyset$ then return { }                                    (30)
else pick $(b, m, g) \in B$                                              (31)
    $\{<\{a \circ b\}, n \cup m, h>\} \cup$ Rewrite ($B - \{(b,g)\}, a, h$)   (32)     □
```

We took our original algorithm and first extended it to account for unions. Here, we make several changes to account for negation. First and foremost, we extended attribution from a pair to a triple consisting of a substitution list (a), a formula (f) to which the substitution list provides a true interpretation, and a set consisting of the attributions for each negated predicate in the formula. As a consequence, the descendants of our initial functions to unfold and rewrite are updated to return triples in lines (23), (29), and (32). More significantly, we must now consider IDB whose definition includes negated predicates as well as negated IDB.

We calculate the attribution of an IDB with negated predicates in line (27). We know that for attributable expressions, the unification of our original formula with the definition of the IDB in line (25) simply adds additional, negated conjuncts. Consequently, we may simply combine attributions for negated predicates in the original expression with attributions for negated predicates in the IDB as seen in line (32).

For negated IDB that are also attributable, we know that certain conditions must hold. Specifically, we know that the IDB must be a disjunction of non-negated predicates. Pushing negations down, this translates into a unifier that effectively substitutes a conjunction of negated predicates for one negated predicate. Accordingly, for each attribution triple of the original formula, we simply remove the attributions for the negated IDB. This is done in lines (15) – (17). In place of these attributions, we substitute the attributions for each predicate in the definition of the IDB. Note that in line (19), we simply attribute the formula for the IDB (assuming the unifier u to avoid conflicts in variable naming). If the IDB is a disjunction, then the attribution will comprise the union of the attributions for each disjunct.

Based upon this revised algorithm, we now offer:

Theorem 4.6 Attribution composition
Our algorithm for attribution composition computes the attribution for attributable expressions.

For IDB that do not include negation, the algorithm is unchanged except for the introduction of a third component to the substitution (which is empty in the case of no negated predicates).

Under this circumstances, the proof therefore follows that of Theorem 1.5. More interesting are the two cases of IDB that include negations and negated IDB.

Our algorithm for attribution composition computes the attribution for the union of attributable, composed CQT$^+$ expressions. Assume the following CQT+ expressions E_1, E_2, E_3 defined by the formulas f, g, and h respectively as:

$E_1 \stackrel{\text{def}}{=} f = (p_1 \wedge p_2 \wedge \ldots \wedge p_n \wedge q) \vee (t_1 \wedge t_2 \wedge \ldots\ )$ where q is the only IDB in E_1
$E_2 = q \stackrel{\text{def}}{=} g = (r_1 \wedge r_2 \wedge \ldots \wedge r_m) \vee (s_1 \wedge s_2 \wedge \ldots \wedge s_o)$ where r_i, $s_i \in d$
$E_3 \stackrel{\text{def}}{=} h = (p_1 \wedge \ldots \wedge p_n \wedge r_1 \wedge \ldots \wedge r_m) \vee (p_1 \wedge \ldots \wedge p_n \wedge s_1 \wedge \ldots \wedge s_o) \vee (t_1 \wedge \ldots\ )$

Note that subject to safety, any of the predicates (with the exception of the IDB q) may be negated. To negate the IDB q, as articulated in Definition 1.17, we are limited to disjunctions of non-negated predicates. Our IDB are thus limited to expressions of the form: $E_4 = q \stackrel{\text{def}}{=} g = r_1 \vee r_2 \vee \ldots \vee r_m$

Given E_1 defined on $d' = d \cup \{q\}$ and r, the result of evaluating E_1 on d', attribution composition computes the [comprehensive | source | relevant] attribution of result r in terms of d as defined by attr(r, E_3, d).

Lemma 4.15 ***$(a_3,h_i) \in A_3$*** **is a comprehensive attribution for *E_3* if and only if *$(a_3,h_i) \in$* Compose *(A_1,f)*.**

Case (→)
We first consider the case where the IDB itself is not negated although any of the base relational predicates (e.g. $r \in d$) may be negated (subject to safety). Pick some $(a_3, n_3, h) \in A_3$. We know that $a_3(f)$ and $a_3(g)$ provide substitutions for the non-negated predicates in a disjunct of f and g by definition. Furthermore, we know that $n_3 = \{(b_1,m,k) | m = \varnothing \wedge k$ is a negated predicate in a disjunct of h that appears also in the corresponding disjunct of $f\} \cup \{(b_2,m,k) | m = \varnothing \wedge k$ is a negated predicate in a disjunct of h that appears also in the corresponding disjunct of $g\}$. For every negated predicate $\neg k$, we know that n_3 includes every substitution b that makes the non-negated predicate k true whether the predicate is in E_1 or E_2. Thus we can model the proof for Lemma 4.10 to verify that the property holds for non-negated predicates and we know that the property holds for n_3, the set of substitutions for negated predicates in h_i.

What then if we allow the IDB q to be negated? We know that to be attributable, the IDB must be defined in the form of E_4, a disjunction of attributable subformulas. Second, we know that when unfolded, pushing down the negation transforms the disjunction into a conjunction where every predicate in g (the formula for E_4) is negated. So in h, by definition for the attribution of negated predicates, n_3 includes the union of the set of all true substitutions for each negated conjunct. But we know that (b_1,m,k) when k is an IDB in E_1 will include every true substitution b_1 for the negated predicate k. Moreover, the negated IDB q is safe in E_1 but were we to attempt attributing the negation of the formula for E_2, we would have an unsafe expression. Instead, we know that q is negated so we call **UnfoldN** instead. In the subsequent call to **RewriteN** we see how we remove the positive substitutions for q (See Algorithm 4.3 line (17)) and replace substitutions in q with the full set of substitutions that make the $u(g)$ (the formula for E_4 subject to appropriate renaming) true (See Algorithm 4.3 line (19)). Thus, we see that n_3 is again $n_1 \cup n_2$ (minus the substitutions for q which do not appear in h_i) and we conclude that (→) holds.

Case (←)

Suppose now that you have some $(a_1 \circ a_2, n_1 \oplus n_2, h_i)$ where h_i is a disjunct of h and $\oplus$ denotes the union of n_1 and n_2 subject to the removal of substitutions for the IDB of n_1 that are unfolded in n_2. (Note that if the IDB is not negated, then $\oplus$ reduces to $\cup$). We pick some $a_1 \in A_1$ and pick a_2 by construction as before. Now, we know that $a_1 \circ a_2$ gives substitutions for non-negated predicates in a_3 as before. However, a_2 now also includes $\{(b_2,m,k)\}$ for negated predicates in g likewise for $\{(b_1,m,k)\}$ in f. But each negated predicate in f and each negated predicate in g is also negated in h by our limitation on attributable expressions. As a consequence, we know that we can $a_1 \circ a_2$ gives a_3 and that $n_1 \oplus n_2$. Hence, we may conclude that (←) holds. □

Lemma 4.16 $(a_3,h_i) \in A_3$ **is a source attribution for** E_3 **if and only if** $(a_3,h_i) \in$ **Compose** (A_1,f) ***w*****here** A_1 **is the source attribution for** E_1**.**

We know by definition that a source attribution does not include substitutions in negated predicates. Therefore, we need only consider the case where predicates other than the IDB are negated. Therefore, we only unfold non-negated IDB and consider only source substitutions in non-negated predicates. We see from Lemma 4.13 that the substitutions in both the negated and non-negated predicates compose. Thus, we conclude that the proof then mirrors the proof for the composition of source attributions for the union of CQT^+ subformulas in Lemma 4.11. In particular, note that ruling out negated IDB, a negated predicate in f or g corresponds to a negated predicate in h and vice versa. Likewise for non-negated predicates. See Lemma 4.11 for the case of a free variable in the IDB. □

Lemma 4.17 $(a_3,h) \in A_3$ **is a relevant attribution for** E_3 **if and only if** $(a_3,h) \in$ **Compose** (A_1,f)**.**

Where A_1 is a relevant attribution for E_1. The challenge in prior classes of queries was to verify that variables relevant in E_1 and E_2 respectively were relevant in E_3 and vice versa. In this way, we could construct relevance in the iterative manner of comprehensive and source attribution. For variables in negated predicates, however, this is trivially true simply because any negated domain variable is defined as relevant. Consider negated predicates in f or g (apart from the IDB). Then, the same variables and predicates are relevant in the unfolding to h and thus relevant. For a negated IDB, our condition on attributable expressions guarantees that every predicate in g is a negated conjunct and is therefore relevant. Thus, for negated

IDB, we simply substitute the negated IDB in (a_1, n, f) with every positive substitution in n. Furthermore, for purposes of safety, every variable in the IDB (q) of f must be relevant in the non-negated predicates and so must also appear in h. □

Thus, building heavily upon Theorem 4.5, we conclude that attribution composition computes the attribution of a composed expression provided that the constituent expressions are attributable. □

4.8 Summary

We began with an overview of the domain relational calculus upon which we build our formal attribution model. We first define our attribution model for simple, conjunctive queries. The model includes definitions for three different types of attribution as well as several different properties of these different attribution types. In particular, we use the properties of conjunctive queries to identify three different categories of equivalence properties and granularity principles.

Having presented a preliminary model, we generalize the model by progressively increasing the expressiveness of the query language for which the model is defined. In the first step, we introduce arithmetic comparisons (omitting explicit equality). Our reliance upon conjunctive query properties to establish equivalence causes conclusions about "relevant" attribution to break down under theta operators. We indicate how explicit equality compromises the attribution of strictly equivalent query expressions.

Subsequent steps introduce union and then negation into the query model. Composition is the only property that continues to hold when unions are permitted. Finally, all attribution properties fail upon incorporation of negation into the query language. However, we define a subset of *attributable expressions* for which the property of composition is preserved.

5 Extended algebra

Unfortunately, while practical systems today are rooted in the Domain Relational Calculus from which we draw our definitions for attribution, conventional systems do not query using the DRC. Fortunately, the relational algebra, a second formal query language that shares the logical foundations of the DRC, aligns closely with SQL, perhaps the most widely used commercial data query language.

In this Section, we operationalize our model by extending the relational algebra to support attribution. We begin by sketching our intuition behind an algebra for attribution. Next, we provide some basic definitions from which we build the extended algebra. After presenting our attribution algebra, we consider some of the extended algebra's properties. We first show that the attribution algebra is closed. We then show that the extended algebra reduces to the standard relational algebra and is a consistent extension of the standard algebra (both properties are elaborated upon below). Finally, we prove that for algebraic expressions without nested negations, the attribution algebra supports the formal model. That is to say that for any algebraic query expression without nested negations, the extended algebra produces the relation-level source granules for attribute-value pairs in the result relation as defined by the formal model.

5.1 Algebra for attribution

In our extended algebra, metadata to calculate source, comprehensive, and relevant attribution is associated with attribute-value pairs of the relational data model. We propagate the attribution metadata in an eager fashion that updates source, relevant, and comprehensive attribution with each successive query operation.

In Section 2 on Related Work, we noted that eager approaches continuously maintain attribution values. While the overhead is higher, response to an attribution request is correspondingly faster. Purely lazy approaches, by contrast, wait until a request for attribution is posed. Depending upon the motivation, different applications might prefer one approach to the other. Because intellectual property provisions, as a matter of policy, apply uniformly, eager approaches may make the most sense. For data sets that are of generally high quality, a lazy approach for tracing anomalous values might be more appropriate.

For simplicity, we leverage the granularity intuition from Section 4. Associating attribution with each attribute-value pair corresponds to value-level result granules. Value-level result granularity preserves the observation that different attribute-values in the same tuple may draw from different sources and be subject to different constraints (source and relevant attribution). Conversely, rather than maintaining substitutions and query expressions, we propagate only relation names and query expressions. Relation-level source granularity certainly does not correspond to all of the different intuitions, but it both limits the amount of metadata maintained and propagated while satisfying the needs for specific attribution motivations. As argued earlier in our discussion of granularity, some issues such as remuneration or intellectual property are addressable by coarse-grained source granules.

5.2 Basic definitions

To present the extended algebra, we begin with a few basic definitions both as a brief review and as an introduction to the notation used throughout the remainder of this Section.

Let $\mathbf{D} = D_1 \cup D_2 \cup \ldots \cup D_n$ be the set of disjoint domains over which all relations are defined. A *scheme* is a pair (J, D) where J is an index (a set of integers) from 1 to $max(J)$ and D is a function that maps every element in the index to a domain in $\mathbf{D}$ $(D : J \rightarrow \mathbf{D})$. Note that in practice, this is no different than traditional attribute-value naming and is done here for notational convenience (Ullman 1988). A *relation* is then defined over a scheme as a finite subset of the Cartesian product of the domains in the scheme. Each element t of a relation R defined on scheme (J,D), written $t \in R$, is a tuple of scalars where for $j \in 1 \ldots max(J)$, $t[j] \in D[j]$.

The *relational algebra* is then defined in terms of two unary and three binary operators that take one (or two in the case of binary operators) relations as arguments and returns a single relation. Domains in $\mathbf{D}$ are considered θ comparable meaning that we can evaluate the binary, Boolean operators $\{<, \leq, =, \geq, >\}$ for values in each domain.

Formal definitions of the unary and binary operators are given below. Here we offer more colloquial intuitions. *Select* (σ) is a unary operator that takes a relation R and a θ-condition. The resulting relation S is a subset of R containing all tuples of R that satisfy the θ-condition. *Project* (π) is a unary operator that takes a relation R on scheme (J,D) and a set of indexes $K \subseteq J$ specifying a subset of the domains in R. The resulting relation S contains unique tuples of R as defined by the projected domains (only values in domains $D[k]$).

Natural Join ($\bowtie$) is a binary operator that concatenates tuples from each input relation R and S to create a single result tuple. For specified attribute domains that appear in both relations (e.g. as in the case of a foreign key), the duplicate occurrence is eliminated. Result tuples are those formed by R and S provided that the tuple from R and the tuple from S agree in the value(s) of all specified duplicate domains. *Union* (*union*) takes two relations R and S, defined on the same schema, and returns a relation containing all tuples in R and S. *Difference* ($-$) takes two relations defined on the same schema and returns those tuples that appear only in R.

Finally, throughout the remainder of this Section we refer to the *source* of a tuple or the source of the specific instance of a value (i.e. the unique tuple in which the referenced instance of a domain value appears) as a scalar representing the relation in which the tuple appears. A *source* is a relation name.

5.3 Extended algebra

5.3.1 Extended relation

We continue to define the set of all domains **D** and a relational scheme (*J,D*) as before. In the standard relation, each relation element is a tuple of scalars drawn from the corresponding domains. In an *extended relation*, however, every scalar is associated with two sets of sources and extended tuples are associated with an additional set of sources.

Definition 5.1 Extended relation (*R′*)
An *extended relation R′* over scheme (*J,D*) is a finite subset of the Cartesian product of cells written $E_1 \times ... \times E_{max(J)} \times 2^S$. □

Definition 5.2 Extended tuple (*t′*)
An element $t' \in R'$ is an *extended tuple* of *R′*. An extended tuple is a tuple of *cells* paired with a set of sources that returns the comprehensive attribution for every cell in the tuple. The j^{th} element of *t′* is the cell denoted by $t'[j]$ and the set of sources comprising the comprehensive attribution for the tuple is referenced as t_C'. □

Definition 5.3 Cell (E_j)
A cell is defined with respect to an extended relation *R′* on a schema (*J,D*). A cell is a triple composed of a scalar drawn from an attribute domain and sets of sources corresponding to the source attribution and relevant attribution for the scalar. For a scheme (*J,D*) and $j \in J$, we call E_j the Cartesian product $D[j] \times 2^S \times 2^S$. We reference these elements as $t_V[j]$, $t_S[j]$, and $t_I[j]$. □

Two or more tuples with identical values but different source sets are said to be weak duplicates. Such tuples are also referred to in the literature on extended algebras as value-equivalent tuples (Dey, Barron, and Storey 1996; Dey and Sarkar 1996).

Definition 5.4 Weak duplicate
Given two extended tuples t_1 and t_2 in extended relation *R* defined over the scheme (*J,D*), we say that t_1 and t_2 are weak duplicates if and only if $\forall j \in J,\ t_{1\ V}[j] = t_{2V}[j]$. □

5.3.2 Operations on extended relations

We now define a number of operations on extended relations from which we will construct our attribution algebra. From these operations we will define our attribution algebra for extended relations.

Definition 5.5 (δ) Weak duplicate elimination
Given an extended relation *R′* defined over the scheme (*J,D*), the removal of weak duplicates in *R′* is a relation over the scheme (*J,D*):

$S' = \delta(R') = \{s | \exists r \in R'$ *and* $s_V[k] = r_V[k],\ s_S[k] = \bigcup_{\forall r\ \in\ dup(r)} r_S[k],\ s_I[k] = \bigcup_{\forall r \in dup(r)} r_I[k]$, *and* $s_C = \bigcup_{\forall r \in dup(r)} r_C \}$. □

Weak duplicate elimination is very much like the coalesce function introduced by Snodgrass (Snodgress 1987 cited in: Bohlen, Snodgrass, and Soo 1996; Dey, Barron, and Storey 1996) to manage value equivalent tuples. Unlike much work in temporal databases, our (δ) is not an algebraic operator that users may use to manage overlapping temporal ranges.[23] Rather, we follow Wang and Madnick (1990) and Dey (1996), where weak duplicate elimination is incorporated into the extension of each algebraic operator's definition (see below) to preserve the relational set semantics, which does not allow weak duplicates.

The reader will note that a similar problem emerges with multiple relations involving the same attribute as in the case of a natural join on a foreign key or attributes used in a θ comparison as in select (σ). Because of the distinction noted previously in Section 4 between natural join on the same attribute domain and θ-comparable attribute domains, we provide for *attribute coalesce*.

Definition 5.6 (κ) Attribute coalesce

Given an extended relation R_1 over the scheme (J,D), a set $K \subseteq J$, the coalesce of R_1 for the attributes in K is the relation $R_2 = \kappa(R_1,L)$ over the domains in (J,D) such that, where $eq(t)$ is the application of the Boolean function verifying equality for all parameters on the values $t_v[k]$ of tuple $t,\ \forall k \in K$:

$R_2 = \kappa(R_1,K) = \{t_2 \mid \exists t_1 \in R_1$ such that $eq(t_1)$ and $\forall j \in J - K,\ t_2[j] = t_1[j]$ and $\forall j \in K$, $t_{2V}[j] = t_{1V}[j]$ and $t_{2S}[j] = \bigcup_{\forall k \in K} t_{2S}[k],\ t_{2I}[j] = \bigcup_{\forall k \in K} t_{2I}[k],\ t_{2C} = t_{1C}\}$ □

Definition 5.7 (σ^+) Select$^+$

Given an extended relation R_1 over the scheme (J,D), a set $K \subseteq J$, and a Boolean function θ over the domain $D(k_1) \times \ldots \times D(k_K)$ the selection of R_1 on the condition θ for the attributes $k \in K$ is $R_2 = \sigma(R_1,\theta,L)$ over the domain (J,D) such that, where $\theta(t)$ is the application of the Boolean function θ on the values $t_V[k]$ of tuple t, we define a function $Relevant(Y)$ that returns the set of variables relevant to the set of domain variables Y and set $X = Relevant(K)$.

$R_2 = \sigma(R_1,\theta,L) = \{t_2 | \exists t_1 \in R_1$ such that $\theta(t_1)$ and $\forall j \in J,\ t_{2V}[j] = t_{1V}[j],\ t_{2S}[j] = t_{1S}[j],\ t_{2C} = t_{1C}$ and if $j \in Relevant(K)$ then $t_{2I}[j] = t_{1I}[j] \cup \bigcup_{k \in K} t_S[k] \cup \bigcup_{k \in K} t_I[k]$ else $t_{2I}[j] = t_{1I}[j]\}$ □

Relevant is recursively defined to identify all sources that are mutually dependent through θ-comparisons. The set I updates which values in the tuple of an extended select are bound by

[23] (Dey, Barron, and Storey 1996) provides a nice review of different coalesce operators in the literature to manage time stamps

evaluating the θ-condition. In this way, we make explicit the observation that the θ-condition is *relevant* to specific values in the corresponding tuple of the result relation.[24]

Definition 5.8 (π^+) Project$^+$

Given an extended relation R_1 over the scheme (J_1,D_1), an index J_2, and a function p from J_2 to J_1, the projection of R_1 w.r.t. p is $R_2 = \pi(R_1)$ over the scheme (J_2,D_2) such that:
$\forall j \in J_2,\ D_2(j) = D_1(p(j))$, and
$R_2 = \pi(R_1) = \delta(\{t_2 | \exists\ t_1 \in R_1$ and $t_{2C} = t_{1C}$ and $\forall j \in J_2,\ t_2[j] = t_1[p(j)]\})$. □

Definition 5.9 ($\times^+$) Cartesian Product$^+$

Given two extended relations R_1, defined over the scheme (J_1,D_1), and R_2, defined over the scheme (J_2,D_2), the Cartesian product$^+$ of R_1 and R_2 is a relation $R_3 = R_1 \times R_2$ over the scheme (J_3,D_3) such that, for $M_1 = max(J_1)$ and $M_2 = max(J_2)$:
J_3 is an index ranging from 1 to $M_1 + M_2$, and
$\forall j \in J_3$, if $j \leq M_1$ then $D_3(j) = D_1(j)$, else $D_3(j) = D_2(j - M_1)$, and
$R_3 = R_1 \times R_2 = \{t_3 | \exists t_1 \in R_1$ and $\exists t_2 \in R_2$ and $t_{3C} = t_{1C} \cup t_{2C}$ and $\forall j \in J_3$, if $j \leq M_1$ then $t_3[j] = t_1[j]$ else $t_3[j] = t_2[j - M_1]\}$ □

Definition 5.10 ($-^+$) Difference$^+$[25]

Given two extended relations R and S defined over the scheme (J,D), the difference of R and S is a relation $T = R - S$ over the scheme (J,D) such that $T = R - S = \{t | \nexists s \in S$ such that $\forall j$, $t_V[j] = s_V[j]$ and $\exists r \in R$ such that $\forall j \in J$, $t_V[j] = r_V[j]$, $t_S[j] = r_S[j]$, $t_I[j] = r_I[j] \cup \bigcup_{\forall s \in S} s_C$ and $t_C = r_C \cup \bigcup_{\forall s \in S} s_C$. □

The set of sources t_I captures our intuition about negation. To verify that some instance of a value (e.g. the value in a specific extended tuple) does *not* exist in some extended relation S', we must compare the value-instance to every valid substitution in S'.

5.3.3 Extended relational operators

Building from the operators defined on extended relations, we can now define the attribution algebra as an extension of the standard relational algebraic operators. The attribution for an expression is then defined inductively from the extended definitions of the operators.

[24] We introduced the function *Relevant* rather than explicitly defining the term because of our difficulty in either explicitly defining the term or in characterizing how tightly our syntactic rule bound the formal definition of relevance. We present the following as one bound on relevance: $relevant(t_S[k])$ is initialized to $\bigcup_{k \in K} t_S[k]$ and recursively defined as $relevant(t_S[k]) \cup t_S[j]$ where $t_S[j] \cap relevant(t_S[k])$ is not empty.

[25] As will be discussed in greater detail below, the treatment of algebraic difference differs from our management of negation in the formal model of Section 4. However, for algebraic expressions without nested negations, we will see that the algebra and the formal model agree.

Definition 5.11 (σ') Extended select
Given an extended relation R', $\sigma'(R',\theta,L) = \sigma^{+}(R',\theta,L)$ □

The extended select is simply the select defined on extended relations.

Definition 5.12 (π') Extended project
Given an extended relation R', $\pi'(R') = \delta(\pi^{+})$ □

The extended project is a projection followed by a weak duplicate elimination in order to account both for duplicates among extended tuples and duplicates among value equivalent tuples.

Definition 5.13 ($\bowtie'$) Extended natural join
Given extended relations R' and S' defined on schemas (J_1,D_1) and (J_2,D_2) respectively with a function p that maps $H \subseteq J_1$ to J_2 such that $D_1(h) = D_2(p(h))$,
$R' \bowtie' S' = \kappa(\sigma(R' \times^{+} S', \theta(=), \forall H), \forall H)$ □

The extended natural join is a Cartesian product on extended relations followed by a selection on equality for all attribute domains used (named) identically as indicated by the function p. Finally, we coalesce on all attribute domains used (named) identically. The reader may observe that the effect of an extended Cartesian product (×') is achieved by taking the extended natural join where H is empty. Likewise, extended Intersection ($\cap'$) is simulated by taking extended natural join on two relations R' and S' defined for the same schema (J,D).

Definition 5.14 ($\cup'$) Extended union
Given extended relations R' and S' defined on the same schema (J,D), the extended union $R' \cup' S' = \delta(R' \cup S')$ where $\cup$ is the standard set union operator. □

Extended union is simply the standard set union operator that uses weak duplicate elimination to manage value equivalent tuples with different sets of sources.

Definition 5.15 ($-'$) Extended difference
Given extended relations R' and S' defined on the same schema (J,D), the extended difference $R' -' S' = R' -^{+} S'$ □

We can now define attribution in the context of our extended relational algebraic operators. As we define attribution, we informally relate our algebraic definitions to the formal model of Section 4. A formal proof of the relationship between the algebraic definition and the formal model is provided later.

Definition 5.16 Comprehensive attribution
The comprehensive attribution for a scalar $t_V[j]$ in the result of an extended relational algebraic expression E having schema (J,D) is defined as the set t_C. □

t_C is in fact the comprehensive attribution for the entire tuple reflecting the observation from the formal model that when considering relation-level source granules, the comprehensive substitutions that make any value of tuple t in the expression true are the same for every other value in tuple t. Moreover, managing the *difference* operator is actually captured in t_C by construction. This explains Definition 5.15 that updates t_C with the comprehensive attribution for every tuple of the negated relation when evaluating the difference of extended relations R and S.

Definition 5.17 Source attribution
The source attribution for a scalar $t_V[j]$ in the result of an extended relational algebraic expression E having schema (J,D) is defined as the set $t_S[j]$. □

The attribution algebra continuously updates the source attribution for each scalar value in an extended relation by managing the set $t_S[j]$. Note that the source attribution for a value in a tuple is not updated by the extended project or extended union except in the case of weak duplicates. In these instances, weak duplicates represent multiple occurrences of an instance in the same relation (project) or distinct derivations for the same instance (union) as discussed in the formal model. Likewise, source attribution is not updated in the case of natural join except for those values that are drawn from the same (named) attribute domain (i.e. coalesced). In the formal model, we identified this as multiple occurrences of the same variable in different conjuncts representing relational predicates. Note also how the set $t_S[j]$ is not altered in the definition of extended set difference, corresponding to our intuition that a negated sub-query is never a source for a value in the result of the difference.

Definition 5.18 Relevant attribution
The relevant attribution for a scalar $t_V[j]$ in the result of an extended relational algebraic expression E having schema (J,D) is defined as the set $t_S[j] \cup t_I[j]$. □

Notice that the relevant attribution is defined in terms of two sets of sources, $t_S[j]$ and $t_I[j]$. The set $t_I[j]$ is not updated for extended project and extended union except in the case of weak duplicates. Because weak duplicates represent distinct derivations for a given instance of a value in the result, we legitimately include the relevant attribution for each weak duplicate. We see that $t_I[j]$ is always updated when evaluating the extended difference but only selectively updated when evaluating θ-conditions.

For extended difference, $t_I[j]$ is updated with the *comprehensive attribution* of *every* tuple in the negated relation. Comprehensively attributing every tuple corresponds to our intuition from the formal model about evaluating the truth of a negated sub-formula. We see that relevant attribution includes $t_S[j]$ corresponding to the idea that the source of a value is certainly relevant.

In the selection operation, we update the relevant attribution for every value in a tuple with the relevant attribution of the selection variables. Intuitively, a selection condition restricts a subset of (possibly all) values in the result tuple hence the introduction of the *relevant* function which relations are linked through θ-comparison. Recall also the implicit selection-on-equality in the natural join. Note that in the special case of natural join where there are no shared variables (i.e. no implicit selection), the relevant attribution for values in the result are drawn exclusively from the corresponding constituent tuple of the Cartesian product. This corresponds to our intuition from the formal model that restricting the tuples in one argument of a Cartesian product is not a restriction on the second argument.

5.4 Properties of the algebra

Having presented our attribution algebra, we now consider properties of the extended algebra. We demonstrate first that the algebra is closed. Then, following the literature on extended algebras for temporal databases (Dey, Barron, and Storey 1996), we establish that the attribution algebra both reduces to and is a consistent extension of the standard relational algebra. Finally, we show that, for a limited set of extended algebraic query expressions, the attribution returned by the algebra corresponds to the relation-level source granules defined by the formal model.

5.4.1 Closure of the extended algebra

The intuition behind closure is that an extended algebraic operation, when applied to an extended relation(s), returns an extended relation. Maier (1983) identifies three requirements:

1. the values in each cell of the extended relation all come from the correct domains
2. there are no (weak) duplicates in an extended relation
3. the relations must be finite

Lemma 5.1 The values in each cell of the output from an extended operation on extended relation(s) all come from the correct domains.

Case ($\pi'(R')$ where R is defined on schema (J,D)): We know by definition that $\pi'(R')$ is defined on a schema (K,D) where $K \subseteq J$ and that for every $s' \in \pi'(R')$ $\exists t' \in R'$ such that $\forall k = p(j) \in K$, $s'_V[p(j)] = t'_V[j]$ so all values come from valid domains. In cases where there are no weak duplicates, then $s' \in \pi'(R')$ is value equivalent to exactly one tuple $t' \in R'$. In this case, $s'_C = t'_C$ and $\forall k = p(j) \in K$, $s'_S[p(j)] = t'_S[j]$ and $s'_I[p(j)] = t'_I[j]$ so the sets of scalars all come from the appropriate domains. If there are weak duplicates among the $t'_V[j]$ for all $j \in K$, then the sets s'_C, s'_S, and s'_I are simply the union of the constituent weak duplicates and the union of valid scalar sets is surely still in 2^S.

Case ($\sigma'(R')$): We assume that R' is an extended relation. Therefore, we know that $t' \in \sigma'(R')$ → $t' \in R'$ so if R' is an extended relation, then $\sigma'(R')$ must also.

Case (($R' \cup' S'$) where R and S are union compatible in the standard sense on schema (J,D)): We know that an extended tuple $t' \in R' \cup' S'$ must come from R', from S', or from both. Consider first the case where t' comes from only one. Then we know for such a tuple t', $\exists r' \in R'$ or $\exists s' \in S'$ such that $t' = r'$ or $t' = s'$ and all values come from appropriate domains. In the case that t' comes from both, then we know, as with weak duplicates in project, that $t'_C = r'_C \cup s'_C$ and $\forall j \in J,\ t'_V[j] = r'_V[j] = s'_V[j]$, $t'_S[j] = r'_S[j] \cup s'_S[j]$ and $t'_I[j] = r'_I[j] \cup s'_I[j]$.

Case ($R' \bowtie' S'$): Recall from the definition that this is a Cartesian product followed by a selection and a coalesce on the common attributes $K \subseteq J$. Certainly the Cartesian product of extended relations is an extended relations because it is merely the $r' \circ s'$ for every $r' \in R$ and $s' \in S'$. Likewise, the select also returns an extended relation (see above). Consider, then, the Coalesce. $\forall t \in R' \bowtie' S'$, t_C is unchanged from the Cartesian product and select. For indexes $j \notin K$ we know that $t[j]$ is unchanged from the Cartesian product and select. For index in K, we know that $t_V[k]$ is unchanged and that $t_S[k]$ and $t_I[k]$ is the union of all values in K where each t_S and t_I is from the correct domains. Hence the union must still be in 2^S.

Case (($R' - S'$) where R and S are union compatible in the standard sense on schema (J,D). For $t' \in (R' -' S')$, $\exists r' \in R'$ such that $t'_C = r'_C \cup \bigcup_{\forall s' \in S} s'_C$ so surely t'_C is from the correct domain. Moreover, $\forall j \in J$, $t'_V[j] = r'_V[j]$ and $t'_S[j] = r'_S[j]$. By construction, $t'_I[j]$ is the union of valid source sets, hence we conclude that the values in each cell of the output from an extended operation on extended relation(s) all come from the correct domains. □

Lemma 5.2 There are no (weak) duplicates in the output of an extended operation on extended relation(s).

First, we know that extended relations are defined as sets so that there are no duplicate extended tuples in an extended relation. A different question is whether the extended operators can produce weak duplicates. We know from their definitions directly that extended select, extended join, and extended difference cannot produce weak duplicates assuming that the initial input relation(s) are valid extended relations (i.e. with no (weak) duplicates). The remaining operators, extended union and extended project both are defined as explicitly calling weak duplicate elimination. Hence, we are assured that there are no (weak) duplicates in the output of an extended operation on extended relation(s). □

Lemma 5.3 The result of an extended operation on extended relation(s) is finite.

Case ($\pi'(R')$ where R' is defined on schema (J,D)): We know that $|\pi'(R')| \leq |R'|$ because each extended tuple of $\pi'(R')$ is a tuple of R' on (K,D) where $K \subseteq J$. At most, every tuple of $\pi'(R')$ is distinct, reduced by weak duplicate elimination. Therefore if R' is finite, $\pi'(R')$ must also be finite.

Case ($\sigma'(R')$): By definition, $\sigma'(R') \subseteq R'$ therefore $|\sigma'(R')| \leq |R'|$.

Case (($R' \cup' S'$) where R' and S' are union compatible in the standard sense): It must be the case that $|R' \cup' S'| \leq |R'| + |S'|$. If R' and S' are both finite, then so is $R' \cup' S'$. Note as in the case of extended project, weak duplicates will reduce the cardinality of $R' \cup' S'$.

Case ($R' \bowtie' S'$): This is a Cartesian product followed by a select and a coalesce. As observed above, an extended select either leaves the cardinality of the input relation unchanged or reduces it. Coalesce merely collapses duplicate attribute (domains); the output of a coalesce has the same cardinality as the input. Thus, we conclude $|R' \bowtie' S'| \leq |R'| \times |S'|$.

Case (($R' -' S'$) where R' and S' are union compatible in the standard sense): Thus $R' -' S' \subseteq R'$ so $|R' -' S'| \leq |R'|$. □

Theorem 5.1 The attribution algebra is closed.
From Lemmas 5.1-3, we conclude that Theorem 5.1 holds. □

5.4.2 Relationship between the standard algebra and extended algebra

Having verified that we can compose operators, we next verify that the extended algebra is both a consistent extension of and reduces to the standard algebra. When we say that the extended algebra reduces to the standard algebra, we are saying that the extended algebra preserves the relational semantics. In other words, from the perspective of the scalar values drawn from attribute domains, the extended operators treat an extended relation on schema (*J,D*) as the standard relation would treat the corresponding standard relation on the same schema and for the same attribute-value substitutions. Following Dey (1996; 1996), we first define a helper function *Reduce*. The purpose of *Reduce* is to take an extended relation and map it to the equivalent relation without the attribution extension. We then show that the extended algebra reduces to the standard algebra through an equivalence proof. The equivalence proof is illustrated in Figure 5.1.

Definition 5.19 Reduce
Given an extended relation R' on a schema (*J,D*), *reduce*(R') = $\{t_2 \mid \exists t_1 \in R'$ and $\forall j \in J, t_2[j] = t_{1V}[j]\}$ also on scheme on (*J,D*).

$$\begin{tikzcd}
 & & S'\,\theta'\,R' \\
S' \arrow[r, dashed] \arrow[d, "Reduce(S')"] & R' \arrow[r] \arrow[d, "Reduce(R')"] & \theta'(R') \arrow[d] \\
S \arrow[r, dashed] & R \arrow[r] & \theta(R) = Reduce(\theta'(R')) \\
 & & S\,\theta\,R = Reduce(S'\,\theta'\,R')
\end{tikzcd}$$

Figure 5.1 Reduction

Theorem 5.2 The extended algebra reduces to the standard algebra

To prove the theorem, we need simply show that the reduction holds for every unary and binary operator of the extended algebra. In each case, we need to show both directions. The reduction of a tuple $t' \in$ extended operator is in a standard operator applied to reduced inputs and vice versa.

Case ($\pi'(R')$ where R' on schema (J_1,D_1)): By definition of *Reduce* we know that R is also defined on (J_1,D_1) and by definition of extended project, we know that $\pi'(R')$ is defined on a function p and produces a schema (J_2,D_2). Note that $\pi(R)$ is defined similarly for R on (J_1,D_1) and the same p. Assume that $R = Reduce(R')$. Pick some $t \in \pi(R)$. Then by definition of π, $\exists\ t_1 \in R$ s.t. $\forall j \in J_2,\ t_1[p(j)] = t_2[j]$. Because t_2 is a set, we know that there may be more than one such t_1, but there is certainly at least one. From the definition of *Reduce*, we know that for t_1 on (J_1,D_1), $\exists\ t_1' \in R'$ such that $\forall j \in J_1,\ t_1'_V[j] = t_1[j]$. But then $\pi'(R')$ must give t_2' on J_2,D_2 where $\forall j \in J_2,\ t_2'[j] = t_1'[p(j)]$ by definition of π'. And because $t_1 = Reduce(t_1')$, certainly $Reduce(t_1'[p(j)]) = t_1[p(j)]$. This tells us that $Reduce(t_2') = t_2$ so $t_2 \in Reduce(\pi'(R'))$. Likewise, pick some $t_2' \in \pi'(R')$ where we know $Reduce(t_2')$ gives t_2 on (J_2,D_2) when $\forall j,\ t_2[j] = t_2'_V[j]$. By definition of π' we know $\exists t_1' \in R'$ such that $\forall j \in J_2,\ t_1'[p(j)] = t_2'[j]$ where there may be more than one such t_1 on (J_1,D_1). But $Reduce(t_1') = t_1$ on $(J_1,D_1) \in R$ where $\forall j \in J,\ t_1'_V[j] = [j]$. This means that $t_1[p(j)] = t_1'_V[p(j)]$ or that $t_2 = t2'_V\ \forall j \in J_2$.

Case ($\sigma'(R')$ where R' is defined on schema (J,D)): Pick $t \in \sigma(R)$ and assume $\neg\exists t' \in \sigma'(R')$ for which $t = Reduce(t')$. We know that $R = \{t \mid \exists t' \in R'$ and $\forall j\ t[j] = t'_V[j]\}$ so for every t, there must be some t'. But t satisfies (θ,L) which means $\forall k,\ t_V'[k]$ also satisfies θ, a contradiction. Now pick $t' \in \sigma\ '(R')$ where $t = Reduce(t')$. Then assume $\neg\exists t \in \sigma(R)$. But if $t' \in \sigma'(R')$ then t'_V satisfies (θ,L). But $\forall j \in J,\ t'_V[j] = t[j]$ by definition of reduce so t must also satisfy (θ,L) which means that $t \in \sigma(R)$, a contradiction.

Case ($(R' \cup' S')$ where R' and S' are union compatible in the standard sense): Pick $t' \in R' \cup' S'$. If we $\mathit{Reduce}(t')$ we get t where $\forall j \in J$, $t[j] = t'_V[j]$. But by definition, we know $t' \in R'$, $t' \in S'$, or both. If $t' \in R'$ then $\mathit{Reduce}(t') = t \in R$ by definition which means $t \in R \cup S$. Likewise for $t' \in S'$ so certainly for both. Now pick $t \in R \cup S$. Then $t \in R$, $t \in S$, or both. When $t \in R$, we know $\exists t' \in R'$ *s.t.* $\forall j \in J$, $t'_V[j] = t[j]$ meaning that $t = \mathit{Reduce}(t')$. But if $t' \in R'$ then $t' \in R' \cup' S'$ and the same for $t \in S$ and again certainly for both.

Case ($R' \bowtie' S'$): Pick $t \in \mathit{Reduce}(R' \bowtie' S')$. Then t corresponds to $t' \in R' \bowtie' S'$ where $\forall j$ $t[j] = t'_V[j]$. (J,D) is the schema for $R' \bowtie' S'$. Then $\forall j$, $t'_V[j]$ is from R' or from S' or from both (if j is in the k's of overlapping domains from which the selection on the Cartesian product is made). But for R', $t'_V[j] = t[j] \in R$. Likewise for S' and S. We note that for $t'_V[k]$, $t[k]$ holds in R and S. Certainly $t \in R \bowtie S$. Now pick $t \in \mathit{Reduce}(R') \bowtie \mathit{Reduce}(S')$. Then $\forall j$, $t[j]$ from $\mathit{Reduce}(R')$, $\mathit{Reduce}(S')$ or both in the event that j is in the k's). From the definition of *Reduce*, we see that $t'_V[j] = t[j]$ in R' and similarly for S'. Finally, for the k's, we see that $t'_V[k] \in R' = t'_V[k] \in S'$. Hence we conclude that $\mathit{Reduce}(R' \bowtie' S') = \mathit{Reduce}(R') \bowtie \mathit{Reduce}(S')$.

Case ($(R' -' S')$ where R' and S' are union compatible in the standard sense. Pick $t \in \mathit{Reduce}(R' -' S')$. Then t corresponds to $t' \in R' -' S'$ where $\forall j$ $t[j] = t'_V[j]$. Then $\forall j$, $t'_V[j] \in R'$ and $\notin S'$. Surely $\mathit{Reduce}(t') = t \in R$. And if $t' \notin S'$ then $\mathit{Reduce}(t') = t \notin S$. So, we know that $t \in \mathit{Reduce}(R' -' S')$ appears in $\mathit{Reduce}(R') - \mathit{Reduce}(S')$. Now pick $t \in \mathit{Reduce}(R') - \mathit{Reduce}(S')$. Then $\forall j$, $t_V[j] \in \mathit{Reduce}(R')$ and $\notin \mathit{Reduce}(S')$. Then $\exists t' \in R'$ such that $\forall j$, $t'_V[j] = t[j] \in R$ and $t' \notin S'$. Thus, we see that $\mathit{Reduce}(R' -' S') = \mathit{Reduce}(R') - \mathit{Reduce}(S')$.

Therefore, we may conclude that for unary operators, $t \in \mathit{Reduce}(op'(R'))$ iff $t \in op(\mathit{Reduce}(R'))$ and for binary operators, $t \in \mathit{Reduce}(R\ op\ S)$ iff $t \in \mathit{Reduce}(R)\ op'\ \mathit{Reduce}(S)$.
□

Having verified that the extended algebra reduces to the standard algebra, we consider the inverse and ask whether the extended algebra is a consistent extension of the standard algebra. In other words, we are asking whether the attribution algebra has the property that every relational algebra expression has a counterpart in the extended algebra. Again following Dey (1996; 1996), we first define a helper function *Extend*. Extend takes an algebraic expression as a single argument and extends the corresponding relation by applying the formal model to the DRC equivalent assuming a database of relations in the original argument. Because there may be more than one valid extended form for a relation (e.g. depending upon the database against which an expression is evaluated), we again turn to an equivalence proof. To demonstrate that the algebra is a consistent extension, we want to show that extending the extended relational operation on the extended relational inputs. This intuition is depicted in Figure 5.2.

Definition 5.20 Extend
Given an algebraic expression E that returns a relation R on schema (J,D), *Extend* transforms E into its DRC equivalent F[26] having formula f to construct the extended relation R'. Let database d be comprised of the relations in the expression E and *granularity* (A) take an attribution and return the relation names corresponding to the substitutions. Then *Extend*(R) = $\{t_2 \mid \exists t_1 \in R$ where $t_{2C} = \mathit{granularity}(\mathit{comprehensive\text{-}attribution}(t_1, F, d)$ and $\forall j \in J$,

$t_{2V}[j] = t_1[j]$

$t_{2S}[j] = \mathit{granularity}(\mathit{source\text{-}attribution}(t_1[j], F, d)$

$t_{2I}[j] = \mathit{granularity}(\mathit{relevant\text{-}attribution}((t_1[j], F, d)\}$ □

$$\begin{tikzcd}
S \arrow[r, dashed] \arrow[d, "\mathit{Extend}(S)"] & R \arrow[r] \arrow[d, "\mathit{Extend}(R)"] & \begin{array}{c} S\,\theta\,R \\ \theta(R) \end{array} \arrow[d] \\
S' \arrow[r, dashed] & R' \arrow[r] & \begin{array}{c} \theta'(R)' = \mathit{Extend}(\theta(R)) \\ S'\,\theta'\,R' = \mathit{Extend}(S\,\theta R) \end{array}
\end{tikzcd}$$

Figure 5.2 Extension

Theorem 5.3 The extended algebra is a consistent extension of the standard algebra
As with reduction, we show that each extended operation is a consistent extension of its standard analog. Let E be an abbreviation for the function *Extend*.

Case ($\pi'(R')$): Pick $t' \in \pi'(E(R)$: By definition, $t'_C = \{R\}$. $\forall j\ t'_V[j] = t[p(j)]$ which is just $\pi(R)$. $t'_S[j] = \{R\}$. $t'_I[j] = \varnothing$. Then certainly $t' \in E(\pi(R))$. Now pick $t' \in E(\pi(R))$. Then $\exists t \in \pi(R)$ for which $t'_V[j] = t[j]$. If we extend t into some t', we know that $\forall j$, $t'_S[j] = \{R\}$, $t'_I[j] = \varnothing$, and $t'_C = \{R\}$. Of course this is just $\pi'(E(R))$.

Case ($\sigma'(R')$ for the selection condition θ,K where R' is defined on schema (J,D)): Recognizing that the selection condition θ,K is the same for both σ and σ', we define the set $X = \mathit{Relevant}(K)$ for both the standard and the extended select. Pick $t' \in \sigma'(E(R))$. By definition, θ,K is true for all t'. Furthermore, we know $\exists t \in R$, $\forall j\ t'_V[j] = t[j]$; $t'_S[j] = \{R\}$; $t'_C =$

[26] See (Ullman 1988)

$\{R\}$ and $\forall x \in X$, $t'_I[x] = \bigcup_{\forall x} t'_I[x] \cup t'_S[x]$. $\forall j \notin X$, $t'_I[j] = \varnothing$.[27] But the set of all such t is simply $\sigma(R)$ which extended is the set of all t'. This is just $E(\sigma(R))$. Similarly, we can pick $t' \in E(\sigma(R))$ which is just the extension of $t \in \sigma(R)$. Then we know that $E(\sigma(R))$ gives t' such that $t'_C = \{R\}$ and $\forall j$ $t'_V[j] = t[j]$; $t'_S[j] = \{R\}$; $t'_C = \{R\}$ and for X (in this case $X = J$), $\forall x \in X$, $t'_I[x] = \bigcup_{\forall x} t'_I[x] \cup t'_S[x]$. $\forall j \notin X$, $t'_I[j] = \varnothing$. But this tuple is certainly in $E(R)$ because $\sigma(R) \subseteq R$ and we know that t satisfies θ, K as does t'. Therefore we know $t' \in \sigma'(E(R))$.

Case (($R' \cup' S'$) where R' and S' are union compatible in the standard sense): If $t' \in E(R) \cup' E(S)$ then $t' \in E(R)$, $t' \in E(S)$, or both. If $t' \in E(R)$ then $\exists t \in R$, $\forall j$ $t'_V[j] = t[j]$; $t'_S[j] = \{R\}$; $t'_I[j] = \varnothing$ and by definition, $t'_C = \{R,S\}$. Certainly if $t \in R$, $t \in R \cup S$. Moreover, because $t' \notin E(S)$ then we know there is no t'' in $E(S)$ for which $\forall j$ $t''[j] = t'[j]$. Thus, we know that $t' \in E(R \cup S)$. The same holds for $t' \in E(S)$. Now suppose $t' \in E(R)$ and $E(S)$. Then $\exists t_1 \in R$ and $\exists t_2 \in S$. If we were to extend t_1 and t_2 we would find that for $t = t_1 \cup t_2$ when $\forall j$ $t_1[j] = t_2[j]$, $t'_V[j] = t_1[j] = t_2[j]$; $t'_S[j] = t_1'_S[j] \cup t_2'_S[j] = \{R,S\}$; and $t'_I[j] = t_1'_I[j] \cup t_2'_I[j] = \{\}$. $t'_C = t_1'_C \cup t_2'_C = \{R,S\}$. But for $t_1 = t_2$ certainly $t \in R \cup S$ hence $t' \in E(R \cup S)$. Now, if $t' \in E(R \cup S)$ then we know that $\exists t \in (R \cup S)$ s.t. $t'_C = \{R,S\}$ and $\forall j$ $t[j] = t'_V[j]$. If $t \in R$ then $t'_S[j] = \{R\}$. Likewise if $t \in S$. If $t \in R$ and $t \in S$ then we know $t'_S[j] = \{R, S\}$. But if $t \in R$ (and not S) then $\exists t_1' \in E(R)$ and there is no $t_2' \in E(S)$ so we know that for $t' = t_1' \cup' t_2'$, $t' \in E(R) \cup' E(S)$. We can say the same if $t \in S$ and not in R. If $t \in R$ and $t \in S$ then we know $\exists t_1' \in E(R)$ and $t_2' \in E(S)$ for which $t' = t_1' \cup' t_2'$. Then $\forall j$ $t_1[j] = t_2[j]$, $t'_V[j] = t_1[j] = t_2[j]$; $t'_S[j] = t_1'_S[j] \cup t_2'_S[j] = \{R,S\}$; and $t'_I[j] = t_1'_I[j] \cup t_2'_I[j] = \{\}$. $t'_C = t_1'_C \cup t_2'_C = \{R,S\}$. Thus, we know $t' \in E(R) \cup' E(S)$.

Case ($R' \bowtie' S'$)[28]: Let R be defined in J_1, D_1 and S be defined on J_2, D_2 with $n = max(J_1)$ and $m = max(J_2)$. The result of the natural join is a relation on scheme J,D where $J \leq n + m$. K is the set of selection attributes where $K \subseteq \{1 \ldots n + m\}$ and p is the projection function for J to $\{1 \ldots n + m\}$. First, assume $K = \varnothing$. Natural join then reduces to Cartesian Product. Pick $t' \in E(R) \bowtie' E(S)$. Then we know that $\forall j_{1..n}$ $t'[j] = t_1'[j] \in E(R)$ and that $\forall j_{n+1..n+m}$ $t'[j] = t_2'[j - n] \in E(S)$. Finally, $t'_C = \{R,S\}$. But then $t'_1 \in E(R)$ corresponds to $t_1 \in R$ and likewise for t_2' and $t_2 \in S$. Thus we see $t \in R \bowtie$ S and $t' \in E(R \bowtie S)$. If $t' \in E(R \bowtie S)$ then we know $\exists t \in R \bowtie S$ s.t. $\forall j_{1..n}$ $t[j] = t_1[j] \in R$ and that $\forall j_{n+1..n+m}$ $t[j] = t_2[j - n] \in S$. But we can extend t_1 to $t_1' \in E(R)$ and likewise for $t_2' \in E(S)$. We construct t' from t_1' and t_2' s.t. $t'_C = \{R,S\}$. Thus, we know $t' \in E(R) \bowtie' E(S)$. Now, we assume $K \neq \varnothing$. We then make use of Theorem 5.1 and the earlier cases for Cartesian product, selection, and then finally projection to verify that the

[27] In this instance, for $x \in X$, $t'_I[x]$ is just $\{R\}$. Otherwise, $t'_I[j] = \varnothing$. Note that in the more general case (as in the inductive case considered later in this Section), the Intermediate set for $t'_I[j]$ of $t' \in \sigma'(R')$ is by default the intermediate value for the corresponding $t'_I[j] \in R'$. As noted earlier, we introduced the function *Relevant* as a proxy for a syntactic rule.

[28] Recall that we define natural join as a Cartesian product followed by a selection on equality for attributes on the same domain, a coalsce, and then a projection of the duplicate columns. If there are no join attributes, then we simply have a Cartesian product. If the two schemas are the same, then we have an intersection.

property holds. In particular, we know that $\forall k \in K$, $t'_S[k] = \{R,S\}$ and that for each $k \in K$, we know that $x \in Relevant(K)$ as in the selection condition. In this instance, $t'_I[x] = \{R,S\}$. For $j \notin K$, $t'_S[j] = \{R\}$ or $\{S\}$ depending upon whether $p[j] \leq n$. Likewise for $j \notin Relevant(K)$, $t'_S[j] = t_1{'}_I[p(j)]$ or $t_2{'}_I[p(j) - n]$ depending upon whether $p[j] \leq n$.

Case (($R' -' S'$) where R' and S' are union compatible in the standard sense): If $t' \in E(R) -' E(S)$ then t' is an extended tuple $t' \in E(R)$ and $t' \notin E(S)$. This means that $\exists t$ s.t. $t \in R$, $t \notin S$, and $\forall j\ t'_V[j] = t[j]$; $t'_S[j] = \{R\}$; $t'_I[j] = \{S\}$; $t'_C = \{R,S\}$. But then $t \in (R - S)$ and it is easy to see that extending t we get $t' \in E(R - S)$. Now pick $t' \in E(R - S)$. Then $\exists t$ s.t. $t \in R$, $t \notin S$ and $t'_C = \{R,S\}$. $\forall j\ t'_V[j] = t[j]$; $t'_S[j] = \{R\}$; $t'_I[j] = \{S\}$. But if $t \in R$ and $t \notin S$, we can extend t to $t'' \in E(R)$ and we know that $t'' \notin E(S)$. It is then easy to see that $t'' = t' \in E(R) - E(S)$.

Therefore, we may conclude that for unary operators, $t' \in E(op(R))$ iff $t' \in op'(E(R))$ and for binary operators, $t' \in E(R\ op\ S)$ iff $t' \in E(R)\ op'\ E(S)$. □

5.4.3 Relationship between the extended algebra and the formal definition

Having related our attribution algebra to the standard relational algebra, we finally consider the relationship between the extended algebra and the formal model of Section 4. In particular, we want to know whether the extended algebra supports attribution as defined in the formal model.

From Theorem 5.2, we know that we can translate query expressions in the extended algebra into equivalent expressions in the standard algebra. From Ullman (1988), we know that we can translate algebraic query expressions into equivalent queries in the Domain Relational Calculus. Therefore, for any query expression in the extended algebra, using the DRC translation of Ullman (1988), we can evaluate whether the relations in the algebraic attribution correspond to the substitutions in the formal model for comprehensive, source, and relevant attribution. The comparison confirms that for algebraic query expressions without nested subtraction in the right hand side of a difference expression (the subtrahend), the algebraic attribution corresponds to the formal model.

We saw in Section 4 that because of its additivity property, attribution has complications when faced with *nested negations* (i.e. $x = \neg(\neg x)$). To account for this limitation, we first verify:

Lemma 5.4 Nested negations
Algebraic query expressions without nested subtraction in the right hand side of a difference expression correspond to Disjunctive Normal Form DRC expressions where negations are pushed down to literals without *nested negations* (e.g. canceling $\neg(\neg x)$). We establish this by induction on the number of operators in the algebraic expression.

In the base case of zero operators, the algebraic query expression is a single relation R on schema (J,D) or a constant relation. We know from Ullman (1988) that this is translated into an equivalent relational predicate $r(X_1,\ldots,X_{max(J)})$ or a corresponding expression for the constant relation $\{t_1,\ldots,t_n\}$ on (J,D) with formula $(X_1 = t_1D(1) \wedge\ldots\wedge X_{max(J)} = t_1D(max(J))) \vee\ldots\vee (t_nD(1) \wedge \ldots\wedge t_nD(max(J)))$ where there is a disjunct for each tuple t_i. Certainly in the base case there are no nested negations.

In the induction hypothesis, we assume that for a query with n operators, assuming no difference operators in the right-hand sub-tree of a difference operator, the resulting DRC translation in DNF with negations pushed down to literals will not nest negations. We want to verify that the same holds for a query expression with $n+1$ operators.

Case $(\pi(R))$: The DRC expression for the projection merely reassigns the set of free and bound variables in the formula for R so that a subset of the free variables in R are free in $\pi(R)$ and all others are bound. Certainly the hypothesis holds.

Case ($\sigma(R)$ where R is defined on schema (J,D)): Without loss of generality, we assume that the selection condition is a single theta comparison on a domain in the schema of R. The formula in the DRC expression for R is f which, by the induction hypothesis, has no nested negations, and the formula for the selection condition is a theta comparison $(X\,\theta\, Y)$, $(X\,\theta\, c)$ or $(c\,\theta\, X)$ where X and Y are variables for domains $D(j_1)$ and $D(j_2)$ and c is a constant drawn from $D(j_1)$. Then, the formula in the DRC for $\sigma(R)$ is $f \wedge (X\,\theta\, Y)$ or $f \wedge (X\,\theta\, c)$ or $f \wedge (c\,\theta\, X)$. If f is in DNF with no nested negations, then we know that we can distribute the conjunction across every disjunct in f without introducing any nested negations.

Case ($(R \cup S)$ where R and S are union compatible in the standard sense): If the formula for the DRC expression of R is f and the formula for the DRC expression of S is g, and by the induction hypothesis, f and g are in DNF with no nested negations when negations are pushed down, then with appropriate renaming and reordering, the formula for the DRC expression corresponding to $R \cup S$ is $f \vee g$. Because f and g are already in DNF, no further distribution is required. Certainly the disjunction of two formulas that satisfy the hypothesis will itself satisfy the hypothesis.

Case $(R \bowtie S)$: The formulas for the DRC of R and S are the disjunctions $f_1 \vee \ldots \vee f_n$ and $g_1 \vee \ldots \vee g_m$ respectively, where any negated literals among the f_i's and g_j's are safe (i.e. bound) within each disjunct. Then with appropriate variable renaming and reordering, the formula for the DRC of $R \bowtie S$ is $f_1 \vee \ldots \vee f_n \wedge g_1 \vee \ldots \vee g_m$. After distribution, we have $f_1 \wedge g_1 \vee f_1 \wedge g_2 \vee \ldots \vee f_2 \wedge g_1 \vee \ldots \vee f_n \wedge g_m$ where each f_i and g_j is a conjunction of positive and negative literals so certainly the formula for the DRC of $R \bowtie S$ is also in DNF where the natural join does not introduce nested negations.

Case ($(R - S)$ where R and S are union compatible in the standard sense where the subtree for S has no difference operators): The formulas for the DRC of R and S are the disjunctions $f_1 \vee$

... $\vee f_n$ and $g_1 \vee ... \vee g_m$ respectively, where any negated literals among the f_i's are safe (i.e. bound) within each disjunct and there are no negated literals among the g_j's. The formula for the DRC of $R - S$ is then $f_1 \vee ... \vee f_n \wedge \neg (g_1 \vee ... \vee g_m)$. Distributing the negation across the disjuncts gives $f_1 \vee ... \vee f_n \wedge \neg g_1 \wedge ... \wedge \neg g_m)$ where each g_j is a conjunction of literals. Distributing the negated conjuncts across the f_i's gives $f_1 \wedge \neg g_1 \wedge ... \wedge \neg g_m \vee f_2 \wedge \neg g_1 .... \vee f_n \wedge \neg g_1 \wedge ... \wedge \neg g_m$. Some of the literals among the f_i's may be negated, but after pushing the negations into the g_j's and further distribution, into DNF, there is no introduction of nested negations.

Consequently, we conclude that for algebraic query expressions without a difference operator in the right-hand subtree of a difference operation, the formula in the corresponding DRC expression, when converted into DNF, will never encounter nested negations when pushing negations down to the literals. □

Knowing that such a relationship between algebraic expressions and DRC formulas holds, we can therefore establish that, for the subset of queries that limits the nesting of difference operators, the attribution constructed inductively in the algebra corresponds to the formal definition.

Theorem 5.4 The attribution algebra corresponds to the formal model where the nesting of difference operators is limited.

As with Lemma 5.1, we establish the theorem by induction on the number of operators in the algebraic expression, comparing the definitions constructed in the algebra to the formal definitions of the corresponding DRC equivalent. For notational convenience, all relations *R*, tuples *t*, and operators σ are implicitly extended.

In the base case of zero operators, the algebraic query expression is a single relation *R* on schema (*J,D*) or a constant relation. We know from Ullman (1988) that this is translated into an equivalent relational predicate $r(X_1,...,X_{max(J)})$ or a corresponding expression for the constant relation $\{t_1,...,t_n\}$ on (*J,D*) with formula $(X_1 = t_1D(1) \wedge ... \wedge X_{max(J)} = t_1D(\ max(J)))$ $\vee ... \vee$ $(t_nD(1) \wedge ... \wedge t_nD(max(J)))$ where there is a disjunct for each tuple t_i.

For a base relation *R* on (*J,D*), we initialize the corresponding sets such that, for tuple $t \in R$, $t_C = R$ and for every *j*, $t_S[j] = R$, $t_I[j] = \varnothing$. Algebraically, then, for $t \in R$:
Comprehensive Attribution for a value $t_V[j]$ is $t_C = R$ for the expression $\pi_{D(j)}(\sigma_t(R))$;
Source attribution for a value $t_V[j]$ is $t_S[j] = R$ for the algebraic expression $\pi_{D(j)}(\sigma_t(R))$;
Relevant attribution for a value $t_V[j]$ is $<t_S[j] \cup t_I[j] = R$ for $\pi_{D(j)}(\sigma_t(R))$.

The corresponding formula for the equivalent DRC is just $r(X_1,...,X_{max(J)})$ so for tuple $t \in R$, the comprehensive attribution for a value $X_i = c_i$ in *t* is the set of substitution lists $\{<c_1/X_1,...,c_{max(J)}/X_{max(J)}>\}$ with no negated substitutions on the expression $\{X_i \mid \exists X_1,\ldots,X_{i-}$

$_1$,X_{i+1}, ...$X_{max(J)}$ $r(X_1, ..., X_{max(J)}) \wedge X_1 = t_1 \wedge ... \wedge X_{max(J)} = t_{max(J)}\}$. Every substitution corresponds to the relation R, which is the attribution t_C in the algebra.

Likewise, the source substitution is just the substitutions in r corresponding to c_i/X_i with no negated substitutions on the same expression as for comprehensive substitution. But the source substitutions for c_i/X_i correspond only to the relation R, which is the attribution $t_S = R$ in the attribution algebra.

Finally, the relevant attribution in the base case is just the source substitution which corresponds to the algebraic definition $t_S[j] \cup t_I[j] = R$, and there are no negated predicates. Thus in the base case we confirm that the attribution algebra corresponds to the formal definitions of attribution.

In the inductive case, as with the relationship between the algebra and the DRC, we consider algebraic expressions with $n+1$ operators.

Lemma 5.5 Inductive case for comprehensive attribution
Case ($\pi(R)$): The DRC expression for the projection merely reassigns the set of free and bound variables in the formula for R so that a subset of the free variables in R are free in $\pi(R)$ and all others are bound. The projection of domains $K \subseteq J$ from scheme (J,D) so that the Comprehensive attribution for any tuple $t' \in \pi(R)$ is $\cup\, t_C \;\; \forall t \in R$ where $t[k] = t'[k]$ for all k (e.g. the weak duplicates t'). From the induction hypothesis we know that t_C corresponds to the substitutions in the equivalent DRC expression. The tuples t corresponding to a weak duplicate of t' are exactly those substitutions that agree in $t[k] = t'[k]$ and make the expression for R true. Therefore, any relation U in t_C corresponds to some substitution for a weak duplicate in the DRC expression for R. Thus we conclude, by the induction hypothesis, that the comprehensive attribution for a value in $\pi(R)$ corresponds to the formal definition.

Case ($\sigma(R)$ where R is defined on schema (J,D)): Without loss of generality, we assume that the selection condition is a single theta comparison on a domain in the schema of R. The algebraic comprehensive attribution for a value of $t' \in \sigma(R)$ is simply $t_C' = t_C$ for $t \in R$ and $\forall j$, $t'_V[j] = t_V[j]$. Likewise, because t' simply denotes the substitutions that make the formula in the expression for R true in addition to making the θ condition true, we know that the substitutions for $t \in R$ are the same substitutions for $t' \in R'$ so the algebraic definition corresponds to the formal model. Moreover, if there were any other substitutions $u \in R$ such that u satisfies θ and $u_V = t'_V$ then $t_V = u_V$ (or else R is not a relation). Thus, we conclude that the comprehensive attribution for a value in $\sigma(R)$ as computed by the attribution algebra corresponds to the formal definition.

Case (($R \cup S$) where R and S are union compatible in the standard sense): For a value in a tuple t that appears only in R or only in S then certainly the algebra and the formal definitions agree given the induction hypothesis that they agreed in R and in S. For a value in a tuple $t' \in R$ and $t' \in S$, the algebra will include t'_C from $R \cup t'_C$ from S. Likewise, the formula in the

DRC is a disjunction $R \vee S$ and will include the substitutions from R and S corresponding to t'. By the induction hypothesis, the substitutions in S correspond to t'_C in S and the substitutions in R correspond to t'_C in R, therefore we conclude that the comprehensive attribution for a value in $R \cup S$ as computed by the attribution algebra corresponds to the formal definition.

Case ($R \bowtie S$): Where K from the select and then coalesce of R and S is empty, a value in a tuple t of $R \bowtie S$ comes either from R or from S but not both. If K is non-empty, then a value in a tuple t of $R \bowtie S$ could come from just R, just S, or both. However, regardless, the comprehensive attribution includes the relations in the comprehensive attribution of R and in the comprehensive attribution of S from the constituents for tuple t, r and s. Moreover, we know that there can only be one such $r \in R$ and $s \in S$ or R and S would not be relations. From the induction hypothesis, r_C and s_C correspond to the formal definition of the comprehensive attribution in R and S respectively. Therefore, every possible substitution that could produce r is reflected in r_C and likewise for s_C. Thus, though t may correspond to multiple permutations of disjunctions from the DRC for R and S, there are no permutations that are not captured in $r_C \cup s_C$, but this is the algebraic construction of the comprehensive attribution for a value in $t \in R \bowtie S$. Therefore, we conclude that the comprehensive attribution for a value in $R \bowtie S$ as computed by the attribution algebra corresponds to the formal definition.

Case (($R - S$) where R and S are union compatible in the standard sense and where the subtree for S has no difference operators): For a value in a tuple t of the difference where $t = r \in R$ and for which there is no s s.t. $r = s \in S$, the attribution algebra will return $r_C \cup_{\forall s \in S} s_C$. Note than any nested difference operators in R are captured in r_C while $\cup_{\forall s \in S} s_C$ captures the intuition of comparing every tuple of S to verify $r \notin S$. The corresponding DRC for R and S are formulas f and g in DNF so that $R - S$ is $f \wedge \neg g$. Distributing $\wedge \neg g$ over the disjuncts of f gives $f_1 \wedge \neg g \vee f_2 \wedge \neg g \vee \ldots \vee f_n \wedge \neg g$. For tuple $t = r \in R$, r_C corresponds to the substitutions in $f_1 \ldots f_n$ such that $t = r$ makes f_i true by the induction hypothesis. Likewise, $\cup_{\forall s \in S} s_C$ corresponds to the set of all substitutions that makes g true. Thus we conclude that the comprehensive attribution for a value in $R - S$ as computed by the attribution algebra corresponds to the formal definition. □

Lemma 5.6 Inductive case for source attribution

Case ($\pi(R)$): Assume $\pi(R)$ is on scheme (J,D) for function p. The DRC expression for the projection merely reassigns the set of free and bound variables in the formula for R so that a subset of the free variables in R are free in $\pi(R)$ and all others are bound. From the induction hypothesis we know that $t_S \in R$ corresponds to the source substitutions in the equivalent DRC expression. The tuples $t \in R$ that produce the weak duplicate $t' \in \pi(R)$ are exactly those substitutions that agree in $t[p(j)] = t'[j]$ and make the DRC expression for R true. Therefore, $\forall j$, any relation U in the set $t_S[j]$ corresponds to some substitution for a weak duplicate in the DRC expression for R. Thus we conclude, by the induction hypothesis, that the source attribution for a value in $\pi(R)$ corresponds to the formal definition.

Case ($\sigma(R)$ where R is defined on schema (J,D)): Without loss of generality, we assume that the selection condition is a single theta comparison on a domain in the schema of R. The algebraic source attribution for a value of $t' \in \sigma(R)$ is simply $t'_S = t_S$ for $t \in R$ and $\forall j$, $t'_V[j] = t_V[j]$. Likewise, because t' simply denotes the substitutions for the free variables in the expression for R such that both R and and the θ condition are true, we know that the substitutions for $t \in R$ are the same substitutions for $t' \in R'$ so the algebraic definition corresponds to the formal model. Moreover, if there were any other substitutions $u \in R$ such that u satisfies θ and $u_V = t'_V$ then $t_V = u_V$ (or else R is not a relation). Thus, we conclude that the source attribution for a value in $\sigma(R)$ as computed by the attribution algebra corresponds to the formal definition.

Case (($R \cup S$) where R and S are union compatible in the standard sense): For a value in a tuple t that appears only in R or only in S then certainly the algebra and the formal definitions agree given the induction hypothesis that they agreed in R and in S. For a value in a tuple $t' \in R$ and $t' \in S$, the algebra will include $t'_S \in R \cup t'_S \in S$. Likewise, the formula in the DRC is a disjunction $R \vee S$ and will include the substitutions from R and S corresponding to t'. By the induction hypothesis, the substitutions in S correspond to t'_S in S and the substitutions in R correspond to t'_S in R, therefore we conclude that the source attribution for a value in $R \cup S$ as computed by the attribution algebra corresponds to the formal definition.

Case ($R \bowtie S$): Where K from the select and then coalesce of R and S is empty, a value in a tuple t of $R \bowtie S$ comes either from R or from S but not both. If K is non-empty, then a value in a tuple t of $R \bowtie S$ could come from just R, just S, or both. Consider the case where the value in t, $t_V[j]$ comes from $r \in R$ or $s \in S$ but not both. First, for any tuple t, we know that there can only be one such r and one such s. From the induction hypothesis, if K is empty or the value does not come from $D_1(k) = D_2(k)$, then it is easy to see that $t_S[j]$ must either be equal to some $r_S[j_1]$ or some $s_S[j_2]$ where R and S are defined on (J_1,D_1) and (J_2,D_2) respectively. If the value does come from some $D_1(k) = D_2(k)$, then algebraically, we know that $t_S[j] = r_S[k] \cup s_S[k]$. In the equivalent formula of the DRC where K is non empty, we know that variable renaming and reordering results in multiple occurences of the same variable name in predicates of R and predicates of S. But every substitution must correspond to predicates of R in $r_S[k]$ and a predicates of S in $s_S[k]$ and none others by the induction hypothesis. Then the source substitutions in the formal model correspond to the algebraic source substitution and we conclude that the source attribution for a value in $R \bowtie S$ as computed by the attribution algebra corresponds to the formal definition.

Case (($R - S$) where R and S are union compatible in the standard sense where the subtree for S has no difference operators): For a value in a tuple t of the difference where $t = r \in R$ and for which there is no s s.t. $r = s \in S$, the attribution algebra will return r_S. The corresponding DRC for R and S are formulas f and g in DNF so that $R - S$ is $f \wedge \neg g$. For tuple $t = r \in R$, r_S corresponds to the substitutions in $f_1 \ldots f_n$ such that $t = r$ makes f_i true by the induction hypothesis. Likewise, the tuple t should not appear in any disjunct of g therefore no substitutions of g should appear as a source for values of t. Thus we conclude that the source attribution for a value in $R - S$ as computed by the attribution algebra corresponds to the formal definition. □

Lemma 5.7 Inductive case for relevant attribution
Case ($\pi(R)$): In the algebra, we project the domains $D_2 \subseteq D_1$ from schema (J_1, D_1). From the induction hypothesis, we know that for any tuple $t \in R$, $\forall j$, $t_S[j] \cup t_I[j]$ returns the set of relation names that contain the substitutions returned by $Relevant(D_1(j))$ in the DRC. Likewise, we know that the DRC for ($\pi(R)$) simply reassigns the free and bound variables in the formula for the expression, which means that in the formal model, the expression is the same so $Relevant(D_2(j_2)) = Relevant(D_1(p(j_2)))$. Thus $\forall t' \in \pi(R)$, the relevant substitutions in the DRC are the same as that for R corresponding to the algebraic definition where $t'_S[j_2] = t_S[p(j_2)]$ and $t'_I[j_2] = t_I[p(j_2)]$. Weak duplicates are simply those substitutions that agree in all of the values of j_2 but not all the values of j_1. But the formal model is a set of substitutions, so for any instance corresponding to the free variables, the substitution is the set of all substitutions that make one instance true and is just the set of all weak duplicates. In the algebra, this is the union of $t'_S[j_2]$ and $t'_I[j_2]$ over all t' that agree in the values $t'_V[j]$.

Case ($\sigma(R)$ where R is defined on schema (J,D)): Without loss of generality, we assume that the selection condition is a binary theta comparison θ, K on a domain in the schema of R. As noted in the definitions earlier, for simplicity, we invoke a function $Relevant(K)$ to return the same domain variables in the algebra as in the DRC expression. Therefore, by the equivalence of $Relevant(X)$ where X ranges over the domain variables in the DRC expression and $Relevant(D(j))$, we see that the algebra begins with the initial relevant relations (induction hypothesis) and incorporates only those relations containing any domain variable X. Hence, we conclude that for $t' \in \sigma(R)$, $\forall j \in J$, the relevant attribution for $t'_V[j]$ corresponds to the relevant substitutions for the set of free variables on the same domain $D(j)$ in the DRC.

Case (($R \cup S$) where R and S are union compatible in the standard sense): For a value in a tuple t that appears only in R or only in S then certainly the algebra and the formal definitions agree given the induction hypothesis that they agreed in R and in S. For a value in a tuple $t \in R$ and $t \in S$, the algebra will combine t_S from $R \cup t_S$ from S and treat the t_I sets similarly (see: weak duplicate elimination). Likewise, the formula in the DRC is a disjunction $R \vee S$ and will include the relevant substitutions from R and S corresponding to the free variables as they appear in relational predicates R and S. By the induction hypothesis, the relevant substitutions in S correspond to $t_S \cup t_I$ in S and the substitutions in R correspond to $t_S \cup t_I$ in R, therefore we conclude that the relevant attribution for a value in $R \cup S$ as computed by the attribution algebra corresponds to the formal definition.

Case ($R \bowtie S$): As in other proofs for natural join, we rely here upon composition and the fact that natural join is defined as a Cartesian product followed by a selection, a coalesce, and a projection. We show that the property holds for natural join with no join variables (Cartesian product), and then rely upon the proofs for selection and projection shown earlier.[29] Every tuple of $R \bowtie S$ is comprised of a tuple $t_1 \in R$ and a $t_2 \in S$. From the induction hypothesis, we know that $\forall j$, *Relevant* in t_1 and *Relevant* in t_2 contains the relations for the substitutions in the corresponding relational predicates of the DRC expression. In concatenating a tuple of R and a tuple of S, certainly the property still holds. Thus, we may continue to apply the induction hypothesis to the subsequent selection on equality and finally project out redundant attributes.[30]

Case (($R - S$) where R and S are union compatible in the standard sense where the subtree for S has no difference operators): For a value $t_V[j]$ in a tuple t of the difference where $t = r \in R$ and for which there is no s s.t. $r = s \in S$, the attribution algebra will return $t_S[j] \cup t_I[j]$ where $t_I[j] = r_I[j] \cup \forall s \in S\, s_C$. In particular, every tuple $s \in S$ becomes relevant because it is used to verify that the instance $t_V[j]$ (defined as the tuple of R containing $t_V[j]$) does not appear in S. r_I is how the substitutions from nested difference operators are carried forward. In Section 4 we spoke of the additivity property in negation and we see the importance here. We account for nested difference operators in the left hand side (minuend) of a difference operator by continuing to add to t_I. Thus we conclude that the relevant attribution for a value in $R - S$ as computed by the attribution algebra corresponds to the formal definition. □

Hence from the base case and Lemmas 5.5 through 5.7, we conclude that when we do not allow nesting of difference operators in the left-hand side of a difference operator, the attribution algebra corresponds to the formal model. □

Particularly interesting about the limitations that we impose on the difference operator is that for such algebraic expressions, the corresponding DRC corresponds to the subset of DRC expressions for which composition holds. Therefore, while the algebra constructs attribution inductively from the leaves of the query tree up to the root, we are equally assured that we can compose attribution by beginning at the root and drilling down to the base relations at the leaves.

5.5 Summary

In this Section, we have presented an extension to the relational algebra that inductively constructs the attribution for value-level result granules in an eager manner, as a part of query processing. Mindful of the potential explosion in the amount of attribution metadata that such a process can create, the algebra manages source granules at the relation level.

We first formalize the relationship between the standard relational algebra and the extended algebra. Subject to some restrictions on the use of negation in query expressions, we then

[29] Note that because we use the function *Relevant* defined to match the formal model in our definition of *extended select* and then explicitly select on equality, the selection variables are by definition relevant to one another and thereby implicitly coalesce the relevant (intermediate) sets.

[30] The reader may recall that in proving the closure of the extended relational algebra, we verified that the result of a Cartesian product on extended relations is an extended relation.

establish that the attribution generated by the extended algebra does correspond to the formal definition as established in Section 4. The relationship between the composition property of attribution and the inductive algebraic process suggests some interesting possibilities for deploying attribution as an accompaniment to a standard query processor or as an external, network service for lazy attribution processing. Moreover, the parameterization of attribution characteristics in the algebra hints at the potential for incorporating either other types of metadata or more complex functions (e.g. data quality) of existing attribution characteristics. We return to these issues in the Conclusion.

6 Attribution and the Web

We began this thesis by hypothesizing an imaginary on-line travel resource integrator that could answer queries not only based upon its own knowledge but also by possibly gathering and utilizing information from any number of unknown sources. Such systems, however, are no longer hypothetical. Integration, whether for travel, finance, healthcare, current events, etc. is now a trademark application of the World Wide Web.

We saw in Section 1 how attribution may serve many different roles in data integration. As a consequence, we identified several dimensions to describe the problem of attribution. Although our initial interest in this thesis stemmed from the Web, Web querying is an active research topic that has only recently begun to approach a uniform standard (Chamberlin et al. 2001a; Fernandez and Marsh 2001). Like the integration that it enables, the underlying theory of Web querying combines several intellectual disciplines including databases, information retrieval, and library science (deBakker and Widarto 2001; Katz 2001; Lenz 2001). As a consequence, we simplified our task by casting the problem of attribution in the context of the relational data model. We presented the formal model in Section 4.

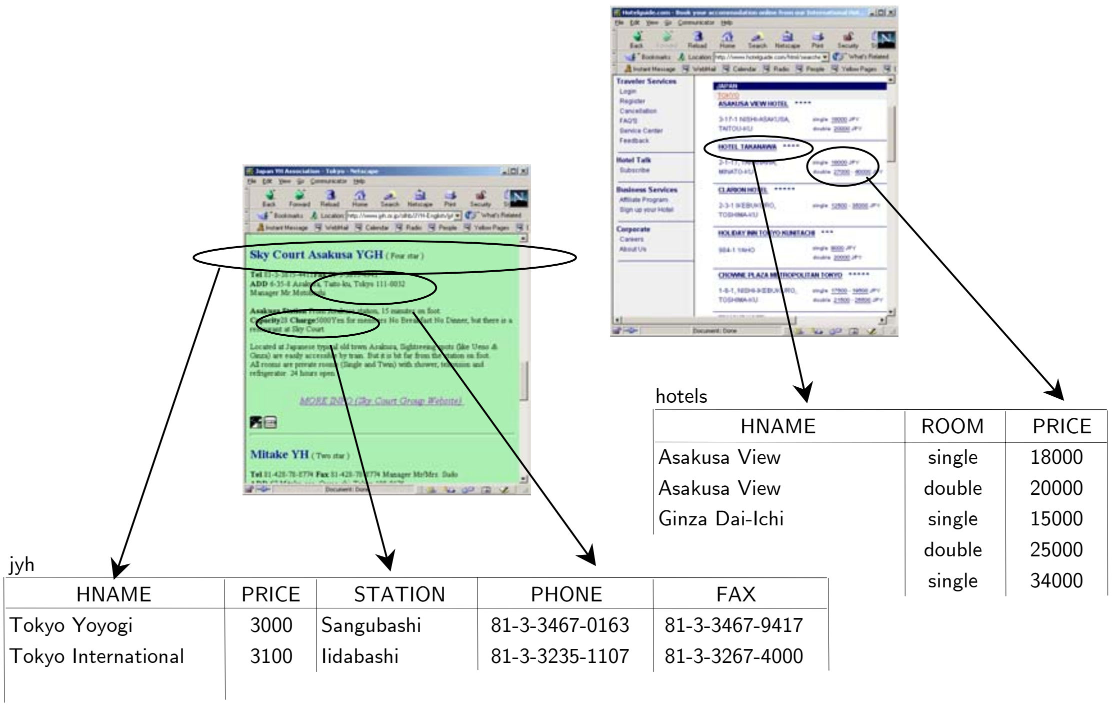
In this Section, we return to the Web. Specifically, we consider how our formal model, developed in the context of the relational data model, relates to the semistructured data model of the World Wide Web. We begin with a very brief overview of some general, semistructured data concepts. Next, we consider how our attribution intuitions from Section 3 relate to the semistructured space. Finally, we consider limitations of applying our formal model of attribution to the Web, referring the reader to work by Buneman et al (2001; 1998; 2000; 2001) on attribution (provenance) for semistructured data.

6.1 Semistructured data models

Research on semistructured data is often confused with evolution of the Web. However, the challenge of data integration existed long before the Web. Current work on semistructured data borrows from portions of the database literature that is often implicitly associated with Web querying: Tsimmis, LORE, Infomaster, Information Manifold (Abiteboul et al. 1997; Chawathe et al. 1994; Duschka and Genesereth 1997a; Duschka and Genesereth 1997b; Levy, Rajaraman, and Ordille 1996). Despite their clear applicability to data on the Web, however, these works were all pursued in the general context of data integration. Indeed, from a data integration perspective, the Web has represented a working infrastructure that simultaneously emphasized the need for and provided a testbed for research on integration and semistructured data (Buneman 1997) (Florescu, Levy, and Mendelzon 1998). In the past five to ten years, interest in and research on semistructured data has exploded. Our goal here is not to summarize the field. Others have covered the foundations (Abiteboul, Buneman, and Suciu 2000). Our goal, instead, is to touch on enough of the basic principles to inform a discussion of how attribution principles might apply in a semistructured environment.

6.1.1 Semistructured data representation

Research in semistructured data models is driven, in no small part, by the observation that data in the "real world" seldom conforms to the well-behaved assumptions that underlie the relational data model. In particular, while data may often be arranged to have the same appearance, the underlying structure or schemas can differ significantly. Consider, for example, the Travel Resource Integrator from Section 1. The travel examples used throughout Part 1 of this thesis draw data from a number of on-line, Web-accessible travel guides. As indicated in Figure 6.1, our initial intuition was to model the data from these Web travel guides as the relations of Section 3.


Japan YH Association - Tokyo - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security
Bookmarks Location What's Related
Instant Message WebMail Calendar Radio People Yellow Pages
Hotelguide.com - Book your accommodation online from our International Hot...
Traveler Services
Login
Register
Cancellation
FAQ'S
Service Center
Feedback
Hotel Talk
Subscribe
Business Services
Affiliate Program
Sign up your Hotel
Corporate
Careers
About Us
Document: Done

Sky Court Asakusa YGH (Four star)

ADD 6-35-8 Asakusa, Taito-ku, Tokyo 111-0032

Located at Japanese typical old town Asakusa, Sightseeing spots (like Ueno & Ginza) are easily accessible by train. But it is bit far from the station on foot. All rooms are private rooms (Single and Twin) with shower, television and refrigerator. 24 hours open.

MORE INFO (Sky Court Group Website)

Mitake YH (Two star)

JAPAN

TOKYO

ASAKUSA VIEW HOTEL ****

HOTEL TAKANAWA ****

CLARION HOTEL *****

HOLIDAY INN TOKYO KUNITACHI ***

CROWNE PLAZA METROPOLITAN TOKYO *****

hotels

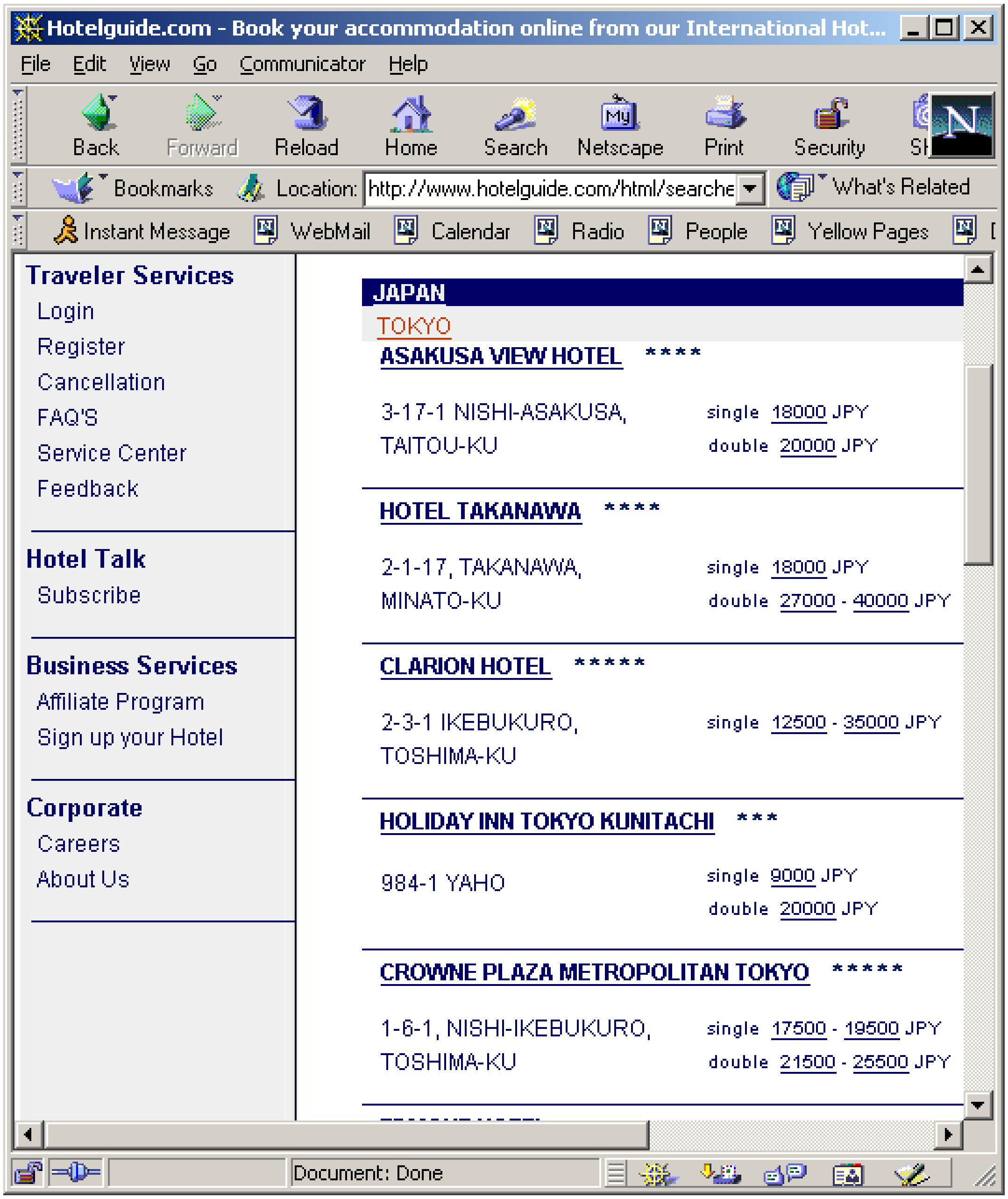
HNAME	ROOM	PRICE
Asakusa View	single	18000
Asakusa View	double	20000
Ginza Dai-Ichi	single	15000
	double	25000
	single	34000

jyh

HNAME	PRICE	STATION	PHONE	FAX
Tokyo Yoyogi	3000	Sangubashi	81-3-3467-0163	81-3-3467-9417
Tokyo International	3100	Iidabashi	81-3-3235-1107	81-3-3267-4000

Figure 6.1 The Web as a relation

Upon closer inspection, however, it quickly becomes apparent that the relational perenity which we assumed in Section 3 breaks down. Consider the Web guide "The Hotel Guide" from which we populated the relation table "hotels" (hotelguide.com 2001). We include one page of hotels in Tokyo, Japan from hotelguide.com in Figure 6.2. Aside from the fact that there are a number of hotels that we omitted simply for tractability reasons, we quickly notice that there are some inconsistencies. Not all hotel listings match the entry for the


Hotelguide.com - Book your accommodation online from our International Hot...
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Sh
Bookmarks Location: http://www.hotelguide.com/html/searche What's Related
Instant Message WebMail Calendar Radio People Yellow Pages
Traveler Services
Login
Register
Cancellation
FAQ'S
Service Center
Feedback
Hotel Talk
Subscribe
Business Services
Affiliate Program
Sign up your Hotel
Corporate
Careers
About Us
Document: Done

JAPAN

TOKYO

ASAKUSA VIEW HOTEL ****

3-17-1 NISHI-ASAKUSA, TAITOU-KU — single 18000 JPY, double 20000 JPY

HOTEL TAKANAWA ****

2-1-17, TAKANAWA, MINATO-KU — single 18000 JPY, double 27000 - 40000 JPY

CLARION HOTEL *****

2-3-1 IKEBUKURO, TOSHIMA-KU — single 12500 - 35000 JPY

HOLIDAY INN TOKYO KUNITACHI ***

984-1 YAHO — single 9000 JPY, double 20000 JPY

CROWNE PLAZA METROPOLITAN TOKYO *****

1-6-1, NISHI-IKEBUKURO, TOSHIMA-KU — single 17500 - 19500 JPY, double 21500 - 25500 JPY

Figure 6.2 Hotels in Tokyo, Japan found in www.hotelguide.com

"Asakusa View Hotel". Some entries, like the "Clarion Hotel" may not quote a price for doubles. Others, like the "Hotel Takanawa" may actually indicate a range of prices by listing two values for a "single". Were the hotelguide.com to treat hotel entries as tuples in a relation, the schema might include the union of all schema elements and set missing values to NULL. Rather than treat these values as explicitly NULL, they are instead simply non-existent. There are certainly other ways in which data on the Web does not conform to the relational model (Florescu, Levy, and Mendelzon 1998). However, our goal here is to motivate the "schema-less" or "self-describing" property that is characteristic of all semistructured data models.

Though there are multiple approaches to semistructured data representations, a common theme in the different representations is an explicit rendering of label-value pairs as a generalization of the "attribute-value" pairs in relations. By explicitly encoding every value with a label, semistructured data models carry structure as a part of the data rather than associating tuples (lists of values) with some external schema that conveys structure and meaning.

The concept of self-describing data is perhaps most easily conveyed in a tree or graph. In this overview, we follow the literature by describing the basic model as an edge-labeled graph where edges one of two categories of information. First, edges may contain typed-data commonly associated with the values in the attribute-value parlance of the relational data model. Second, edges may contain names or scalars that are colloquially associated 7with the "attribute" of an attribute-value pair.

In Figure 6.3, we suggest a semistructured model for two hotel entries from hotelguide.com. The reader should notice how every label or edge is associated with a value where the value may be a data value or a node denoting a set of label value pairs.

Although not depicted here, the basic model for semistructured data allows for the explicit association of a unique identifier with a node in the graph. Object identity provides a convenient mechanism for extending tree-structures, such as those depicted in Figure 6.3, into a graph.[31] The reader may also notice that in our example, there are no values on internal edges. Though not necessary, the basic model does not allow values on internal edges. Whether values are assigned to nodes versus edges and whether values are allowed on internal nodes or edges are all variations on the basic model.[32]

Following (Abiteboul, Buneman, and Suciu 2000), we can serialize our graph using the following grammar. If *s* is a semistructured data expression and *oid* is an object identifier that names a node from which edge(s) depart:

[31] Our existing hotel data might not provide the best opportunity for demonstrating graph extensions. The reader is encouraged to refer to (Abiteboul, Buneman, and Suciu 2000) for examples. The reader may also be familiar with the use of IDREF in XML to serve a similar function (Bray, Paoli, and Sperberg-McQueen 1997).

[32] The reader is encouraged to see (Abiteboul, Buneman, and Suciu 2000) for a discussion of these variations.

<s> ≔ *<value>* | oid *<value>* | oid
<value> ≔ atomic value | *<complex value>*
<complex value> ≔ {label: *<s>*,…, label: *<s>*}

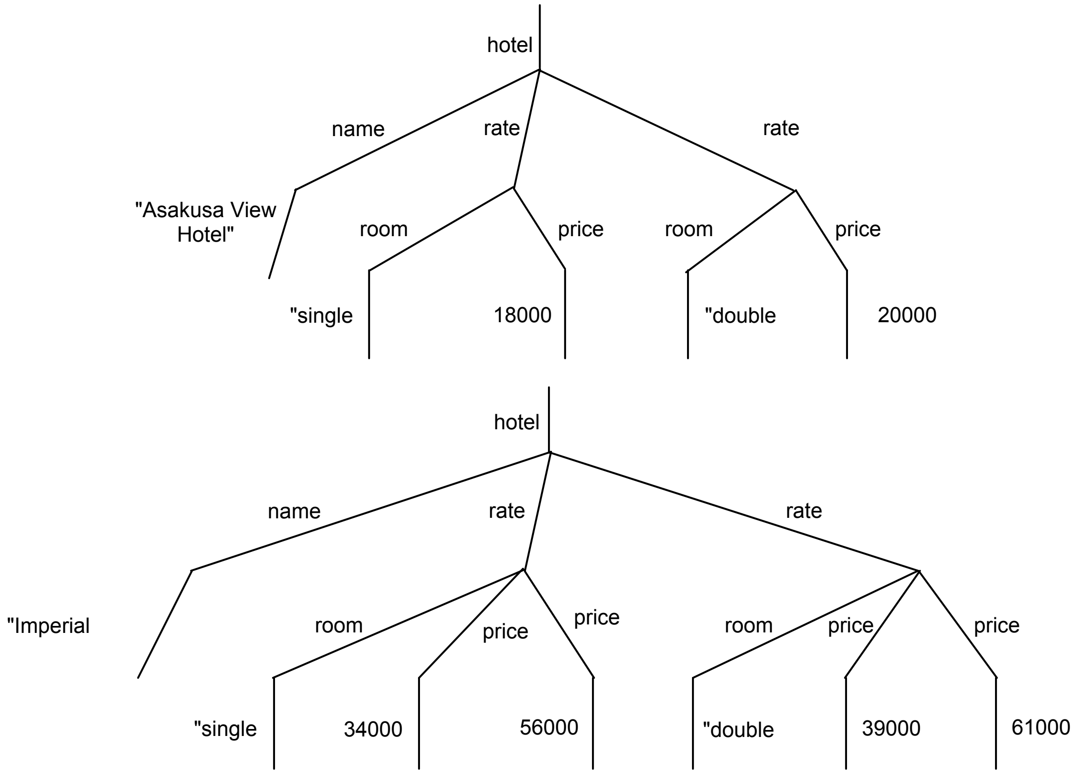


Figure 6.3 Semistructured data from www.hotelguide.com

Example 6.1 Serializing a graph
If we follow convention and name *oid*s using ampersands (e.g. &o1), we can serialize Figure 6.3 as follows:

```
{hotel:  &o1{name:  &o2"Asakusa View",
             rate:  &o3{room:  &o4 "single",
                        price: &o5 18000}
             rate:  &o6{room:  &o7 "double",
                        price: &o8 20000}
           },
…
 hotel:  {name:  "Imperial Hotel",
             rate:  {room:  "single",
                     price:  34000,
                     price:  56000}
             rate:  {room:  "double",
                     price:  39000,
                     price:  61000}
           },
   ...
} □
```

In our serialization of Figure 6.3, we deliberately omitted object identifiers from the second hotel listing. We did so to emphasize the characteristic that, like object-oriented models in general, the basic semistructured data model supports node identity. The model allows for the explicit assignment of a unique identifier to a node. In the absence of assignment, every node has an implicit identifier to establish the uniqueness used in data processing.

6.1.2 Semistructured data manipulation

Query languages serve two fundamental objectives: selection (to avoid confusion with the relational select (σ) operator, we may also use the term "extraction") and presentation. A relational query operator takes one or more relations, each of which is defined on a schema, and extracts some subset of tuples. A new relation is constructed from the extracts. Similarly, operators to manipulate semistructured data take, as arguments, the nodes and edges that constitute one or more graphs. After extracting some subset of nodes (and edges), a semistructured operator constructs a new graph. Just as there are different relational query languages, there are different semistructured query languages. In this subsection, we focus on a few shared concepts for selecting and presenting semistructured data.

6.1.2.1 Data extraction

All semistructured query languages support an elementary form of extraction based upon path expressions. Path expressions are the basic construct with which semistructured query languages specify nodes in a graph. A path is a well-understood concept from graph theory, but we can define a path on semistructured data informally as a sequence of edges between two nodes. The path expression $/l_1/l_2/_{\ldots}/l_n/l_b$ denotes a path from node *a* to node *b* if the graph contains nodes $x_1 \ldots x_n$ and edges such that $(a\ l_1\ x_1), (x_1\ l_2\ x_2), \ldots, (x_n\ l_b\ b)$ (Abiteboul, Buneman, and Suciu 2000). We may then think of a path expression as a query constructor. The result of a path expression applied to a graph is the set of all nodes *b* for which there are edges $l_1, l_{2,} {}_{\ldots}\, l_n, l_b$ from *a* to *b*.

Example 6.2 Path expressions
For example, the path expression `/hotel/name` applied to the graph of Example 6.1 returns the set of nodes for the edges `"Imperial Hotel"`, `"Asakusa View", etc`.

The path expression /hotel/rate/price returns the set of nodes for the edges `18000, 20000, 34000, 56000, 39000, 61000, etc.` □

Path expressions are richer than a sequence of labels, however. By applying regular expressions on the alphabet of edge labels, we expand the paths denoted by (and hence the set of nodes returned by) a single path expression.

Example 6.3 Regular expressions in path expressions
Following the regular expression syntax in Perl, we may write the following path expression: `/(hotel|hostel)/*/price`. Certainly the path: `/hotel/rate/price` matches the pattern of the path expression; among others, the path expression returns the set of nodes for all hotel prices from Figure 6.3. We could also imagine integrating data from the jyh relation with data from www.hotelguide.com by expanding the graph of Figure 6.3 with hostel edges of the form seen in Figure 6.4. Now our path expression also matches the path `/hostel/charge/member/price`. The set of nodes returned by the original path expression now also includes nodes associated with hostel prices. □

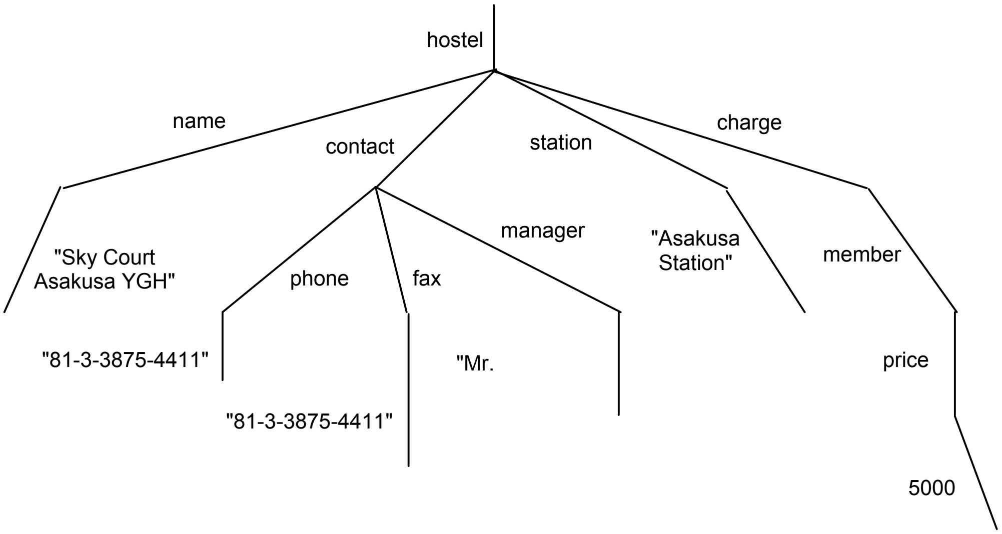


Figure 6.4 Representing hostel Web data in a graph

6.1.2.2 Data presentation

While path expressions return a set of nodes, as a query language, path expressions are incomplete. Path expressions can extract, but a set of nodes does not by itself constitute a graph.[33] We need tools to control presentation (i.e. construct a graph from the nodes in the result of a path expression). The use of variables, in conjunction with path expressions, supports presentation. The result of a path expression is assigned to a variable. These variables are used in the specification of an output path. The output path is a template for the graph of the result of a path expression. In the same way, the "select" clause of an SQL statement defines the schema of the result. Variables and path expressions together complete the basic elements of a semistructured query language. Details of explicit query syntax may

[33] The closure property suggests that, given graphs on inputs, the query language returns a graph.

vary among specific semistructured query languages, but the roles served by variables and path expressions are roughly the same.

Example 6.4 Constructing the result graph of a semistructured query
We use the same path expression as before to extract possible prices for lodging in and around Tokyo, Japan except now we assign node instances to the variable X: `/(hotel|hostel)/*/price X`. Now we build a path as a template for the output of the path expression: `/lodging/price/X`. This path corresponds to the graph of Figure 6.5. □

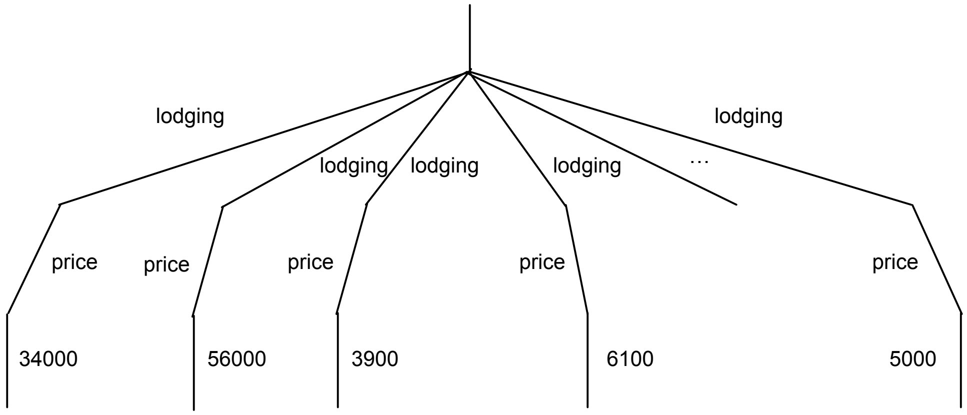


Figure 6.5 Semistructured query result

6.1.2.3 Extending data manipulation capabilities

While path expressions and variables provide the basic infrastructure for a rudimentary, semistructured query language, these tools also support much richer classes of queries. With variable assignment, semistructured languages can support θ-comparisons to further restrict the subset of nodes extracted. Through variable assignments and nested queries, we can support complex graph restructuring.

Example 6.5 θ-comparisons and graph restructuring in semistructured queries
Our query might first consider each hotel or hostel separately.

```
        /(hotel|hostel)/ X
```

A "for-each" conjunction of conditions on every `X` nests one query within another. For each hotel or hostel node, we assign the name to `Y` and the price to `Z`.

```
        /X/name Y
        /X/*/price Z
```

We can apply a boolean test on prices to further restrict nodes in the result graph.

```
        Z < 35,000
```

We then use our variables to define a path as a template for the result graph.

```
        /Y/price/Z
```

The final result graph is depicted in Figure 6.6. □

In introducing semistructured data manipulation, we have deliberately left out many details that we feel are less germane to attribution. Most semi-structured query languages use SQL-like `select-from-where` syntax and some familiar notation for expressing regular expressions on paths.[34] In addition, we can apply regular expressions on labels themselves. For example the conjunction of two path expressions `/(hotel|hostel) X` and `/X/name/"A*"` represent a pattern to get all lodging nodes with names beginning with the letter "A." Other issues involve duplicate management in the face of object identity and the type coercion required to perform θ-comparisons on edge labels or restructure graphs using internal edge labels as values and vice versa. The reader is encouraged to consult other sources on the subject (Abiteboul 1997; Abiteboul, Buneman, and Suciu 2000; Abiteboul et al. 1997; Abiteboul and Vianu 1997; Buneman 1997; Buneman et al. 1997; Buneman, Deutsch, and Tan 1998; Chawathe, Abiteboul, and Widom 1999; Fernandez et al. 1997a; Fernandez et al. 1997b; Lenz 2001).[35]

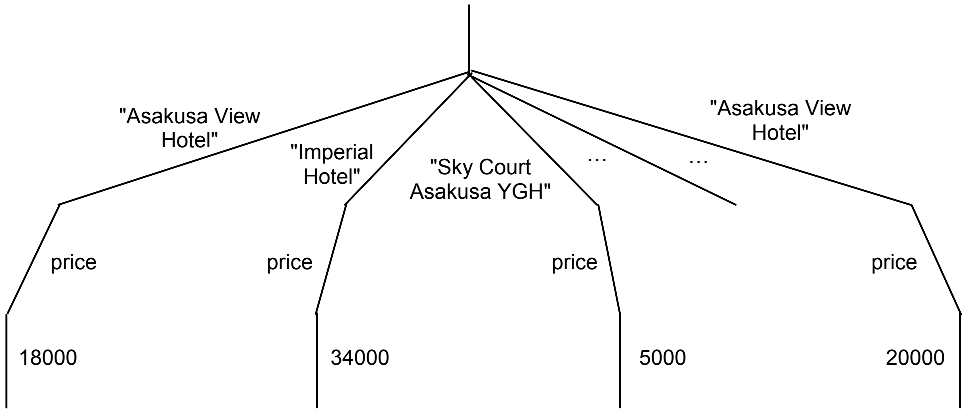


Figure 6.6 Nested queries and complex restructuring

6.2 Attribution intuitions and semistructured data

Having introduced some of the basic principles of semistructured data representation and manipulation, we next consider how some of our attribution intuitions apply to the semistructured context. While the relationship between attribution in the different models is imperfect, in this first section, we consider only how the intuitions do match. In the following section we raise some of the complications.

[34] Familiar notations for paths include "." or "/" separators. Regular expression symbols include "*" for zero or more, "?" for zero or one, "+" for one or more, etc.

[35] We intentionally steered away from explicit reference to XQuery, XPath, and other rapidly evolving World Wide Web Consortium (W3C) standards for querying XML. We did so first to avoid the popular misconception that XML queries are synonymous with rather than simply one (albeit prominent) example of semistructured querying. Second the W3C standards were evolving too rapidly for us to consistently track in this document. We do include references to the W3C work both in the in-text citations above and in the References.

Our general intuition in the formal model of attribution was of substitutions that make an expression true. If we think of a semistructured query as a conjunction of path expressions, the analogy seems simple enough. The attribution for a semistructured query constitutes the subgraphs that match a particular pattern corresponding to the nodes in a result graph.

Example 6.6 Subgraphs that match a particular pattern
In Example 6.3, we gave the following path expression: `/(hotel|hostel)/*/price`. Based upon our sample data from Figures 6.3 and 6.4, we know that the following paths all match the pattern:

```
/hotel/rate/price for the nodes with:
/hotel/name/"Asakusa View" and /hotel/rate/price/18000;
/hotel/name/"Asakusa View" and /hotel/rate/price/20000;
/hotel/name/"Imperial Hotel" and /hotel/rate/price/34000;
/hotel/name/"Imperial Hotel" and /hotel/rate/price/56000;
/hotel/name/"Imperial Hotel" and /hotel/rate/price/39000;
/hotel/name/"Imperial Hotel" and /hotel/rate/price/61000;
```

and

```
/hostel/charge/member/price for the node with
/hostel/name/"Sky Court Asakusa YGH" and /hostel/charge/member/price/5000
```

□

In the formal model, we explored different categories of equivalences. For the concept of strict equivalence, the difference between object-identity and value-equivalence introduces a slight inconsistency, but even with object-identity, we can imagine multiple paths in a graph to the same node.

Example 6.7 Strict equivalence: multiple paths to the same node in a graph
For example, suppose two different youth hostels shared the same manager. We illustrate such a possibility in Figure 6.7. □

The potential for cycles in a graph, of course, will also result in multiple paths to the same node. In the formal model we encountered a related problem posed by the potential introduction of redundant conjuncts. The relational calculus has the concept of a minimal query and the question of a minimal path is an open question that we raise as a challenge below and direct the reader to external references (Abiteboul, Hull, and Vianu 1995).

Equivalence through composition is a second category of equivalence. In the formal model, attribution composition stems from query composition (i.e. using the result of one query as the input to another as in IDB) The principle behind attribution composition is to recursively construct attribution in a step-wise fashion rather than to unfold the entire query a'priori or to carry metadata attribution forward with each value, updating with every additional operator.

Query and attribution composition has particular relevance for semistructured data and the Web in particular. Querying against one or more graphs returns a new graph that itself can serve as a source for a new path expression. Web portals and other aggregation engines serve in this very manner. In Section 1, we recounted the lawsuit between Priceman and MySimon.

We may characterize a page in MySimon as the result of query that itself became a source for Priceman. Analogously, we may compose attribution in a stepwise fashion.

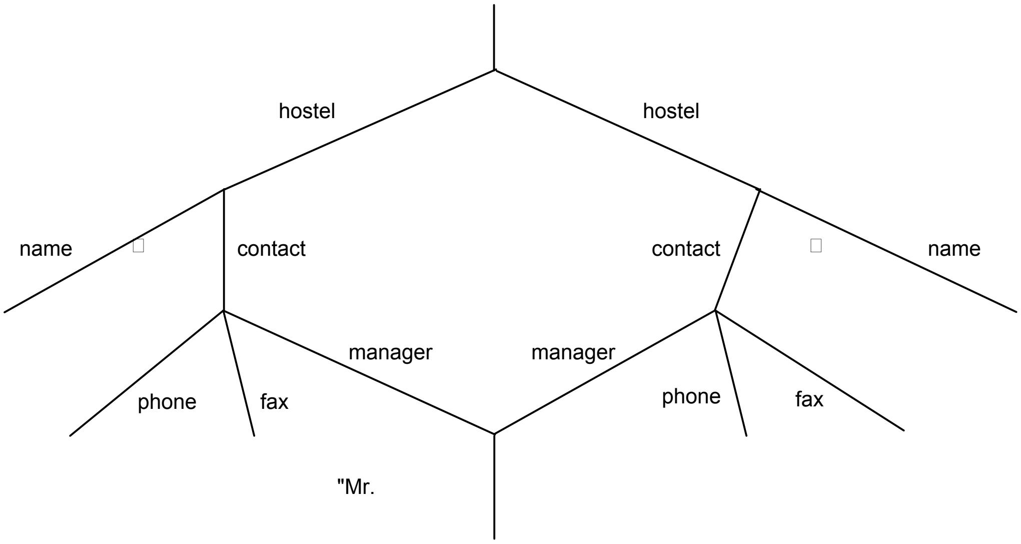


Figure 6.7 Strict equivalence in semistructured data

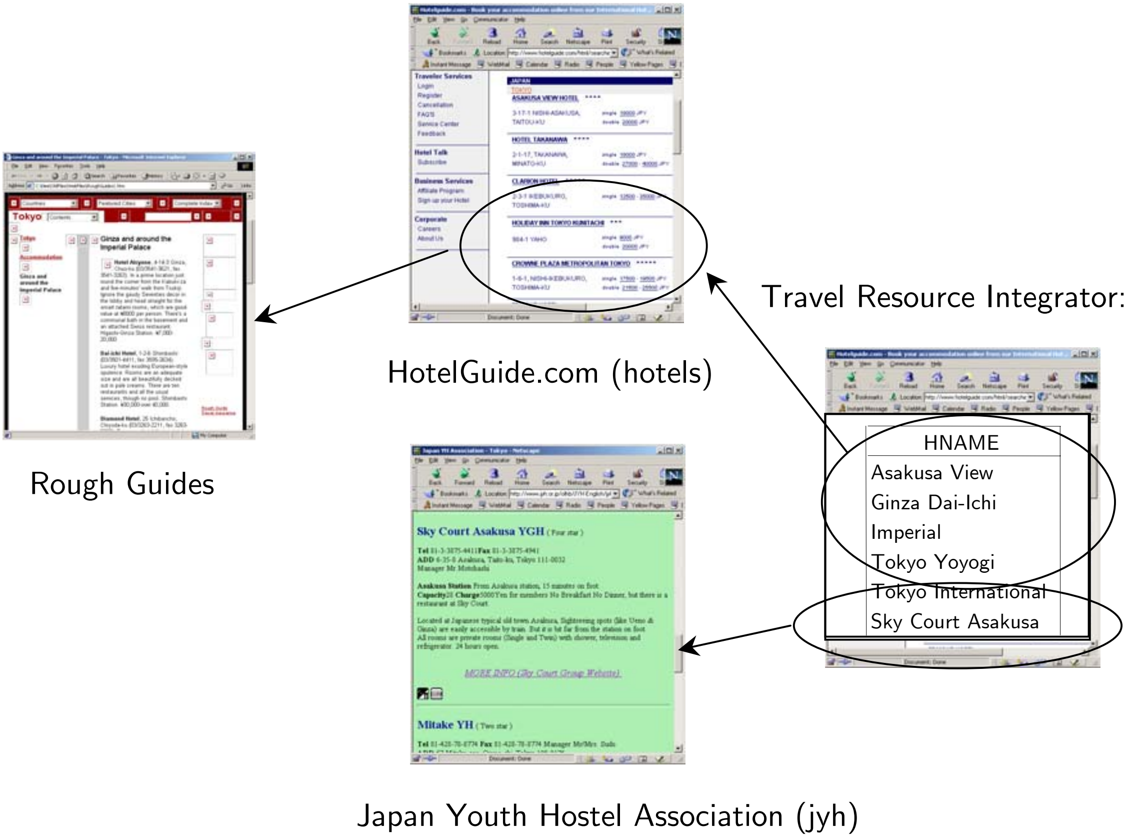
Example 6.8 Attribution composition for semistructured data
From our Travel Resource Integrator of Section 1, we could imagine attributing the result of a query on Tokyo sites to sources including www.hotelguide.com and www.jyh.com. We could equally imagine that these sites might in turn aggregate information from additional sources. Perhaps we might attribute the `"Asakusa View Hotel"` in www.hotelguide.com and discover that the listing was itself extracted from a RoughGuides travel guide as in Figure 6.8.[36] □

Finally, we consider our observations from Section 4 on coarse- and fine-grained source and result granularity. Our intuition for result granularity was the thought of rolling-up attribution from a value to its identifying tuple or to its domain. Likewise, a domain or a tuple may share attribution characteristics with the containing relation. Attribution at a higher level of result granularity aggregates the attribution for each constituent. Source granularity combines our ideas about result granularity and composition. Recognizing that a result granule associates attribution with some subset of values, and that the result granule can itself constitute a source for a composed query, we arrive at the concept of a source granule. Rather than attributing from substitutions in a source relation, we might attribute to source tuples or source relations.

In Example 6.5, we saw how we could use a path expression to reference an internal node. As with our example, at least some semistructured query languages use references to internal nodes as a form of syntactic sugar for nesting queries on the independently named sub-elements (Abiteboul, Buneman, and Suciu 2000). Accordingly, we might envision using this

[36] hotelguide.com does not indicate that it uses Baedekker's as an external reference and we use the example here merely to illustrate the concept of query and then attribution composition.

notation to associate attribution with some internal node, implicitly referencing the subgraph rooted at the internal node. Attribution to an internal node would correspond to the idea that coarse granularity captures the attribution for each constituent. Moreover, because path expressions constitute query selection constructors, we can think of specifying arbitrary granules with query expressions. Colloquially, we can talk about attributing parts of a Web page rather than the page en masse as in a bibliography or individual items as in a footnote. Indeed it was because of observations about granules in semistructured data that we sought an analogy in the relational context.


Rough Guides
HotelGuide.com (hotels)
Travel Resource Integrator:
Japan Youth Hostel Association (jyh)

HNAME
Asakusa View
Ginza Dai-Ichi
Imperial
Tokyo Yoyogi
Tokyo International
Sky Court Asakusa

Figure 6.8 Attribution composition in semistructured queries

Example 6.9 Granularity for semistructured data attribution

Referring again to Figure 6.8, we indicate how a result may be separated into different granules. Hotel information comes only from HotelGuide.com and likewise for hostel information. Similarly, we may not have used all of the information from HotelGuide.com, so we can separate their data into source granules. If information about a hostel's address information comes from a different place than pricing and management information, then we may think of each source as the result of a query on some other source and attribute accordingly. □

6.3 Challenges for attributing the Web

While many of our intuitions appear to map in a straightforward manner to the Web, there are a number of confounding factors that make attribution on the Web a challenge in its own right. First and foremost is the recognition that the Web itself does not conform to our basic, semistructured data representation. As a consequence, we separate our discussion of challenges first into issues posed by semistructured data in general and then Web specific concerns.

6.3.1 Challenges attributing semistructured data

First, we note that in the formal model, attribution is modeled as external metadata set apart from data domains and relations on domains. Accordingly, in the algebra, we extended the data model by associating metadata with values and tuples (Sadri 1991) rather than incorporating attribution metadata explicitly into the relational schema (Dey, Barron, and Storey 1996; Dey and Sarkar 1996). In the semistructured context, it is easy to see how attribution could emerge as a metdata graph rooted to every node in the data graph through an "attribution label." Changes in query semantics as well as implications for a data representation that essentially duplicates paths in the graph need further thought.

Second, our intuitions about query composition and attribution composition, while analogous to their relational counterparts in the abstract, suffer from some difficulties in the details. In the formal model, the attribution for a composed query is defined by the attribution for the unfolded calculus expression. However, it is not clear that there is an equivalent for unifying two semistructured path expressions. Consider the case where one expression is a restriction and restructuring of the same graph (i.e. using the same nodes and labels).

A related problem, alluded to earlier, is the issue of recursive queries. Given a finite graph to begin with, we know that a path expression on a finite graph, recursive or otherwise, must return a finite set of nodes. However, the explicit paths that we can associate with the path expression for any given node, in the presence of a cycle, can be (countably) infinite. While there has been recent work on recursive queries in semistructured data (Abiteboul, Buneman, and Suciu 2000), finding a corresponding resolution for attribution will require some additional consideration.

Finally, graph reconstruction also poses a problem for our intuitions about granularity and aggregating attribution over subgraphs. Because variable assignment allows unrestricted (re)use of a given node (label) in structuring a result graph, there is no necessary dependency between the attribution of a node and the attribution of its children in a graph. For example, we could imagine constructing a result graph that associates hostel prices with hotel nodes. The attribution of the hotel node would have no bearing on the attribution of the hostel prices.

6.3.2 Challenges attributing the Web

Other challenges derive from the nature of the Web itself and the recognition that data on the Web today does not correspond neatly to any formal semistructured model. First, we know that the relational data model is value oriented. Every tuple instance is unique by definition. As noted earlier, in semistructured data, object identity causes value-oriented uniqueness to break down. On its own, this does not pose a problem, as the concept of identical values with different attribution appears in the formal model. A related problem that does emerge, however, is the question of duplicates. More generally, reflecting the Web's document-centric history, every node is represented (no weak duplicates), and order matters (Bray, Paoli, and Sperberg-McQueen 1997).[37] The need to reference order, both for querying and attributing, requires richer concepts.[38]

Apart from label order, the labels themselves pose some difficulty for being able to construct precise paths. Although standards for XML, in conjunction with XSL and Style Sheets, are evolving to address issues of meaningful, content-based labels, the Web today is dominated by HTML (Chamberlin et al. 2001b; Clark and DeRose 2001; Fernandez and Marsh 2001; Fernandez and Robie 2001; Grosso and Walsh 2000; Raggett 2000). Without special knowledge, then, there is a limit to the data that we can extract and attribute. Consider again hotelguide.com and their Web page on Tokyo hotels in Figure 6.2. While we hypothesized a semistructured representation in Figure 6.3, the data on the page really appears as the HTML that appears (in a slightly abbreviated form) in Example 6.10.

Example 6.10 HTML for hotelguide.com
A vision for the very near future of the Web calls for servers that return XML pages associated with style sheets to control presentation (Bray, Paoli, and Sperberg-McQueen 1997). Today, however, most sites, like hotelguide.com, still present HTML. Excerpted below is edited source from the page for Figure 6.2.

```
<body>
   <table width="100%" cellpadding="0" cellspacing="0" border="0">
   <tr>
      <font class="hotellist"><b>
      <a href="/html/searchengine/">
         ASAKUSA VIEW HOTEL
      </a></b>
   </tr>
   <tr>
      <td width=32% valign="top">
         <font class="hotellist"><font size="1">
            3-17-1 NISHI-ASAKUSA, TAITOU-KU
         </font></font>
      </td>
      <td width=24%>
         <font class="hotellistsmall">
```

[37] The Web (and HTML in particular) was originally conceived as a tool for sharing research literature. As a consequence, particular attention was directed towards formatting and presentation. So while an academic paper is, abstractly, composed of different sections, we might wish to ensure that "Section 6 Data analysis" comes after "Section 1 Introduction." Notice that our graph-based basic semistructured representation has no provisions for explicitly stating that one node or label is first in a sequence.

[38] The reader may note that the problem is not "duplicates" per say but rather one of "order." We merely use duplicates as an example of the need to define explicit order.

```
            single
            &nbsp;
            <a href="JavaScript: newWindow=currencyconverter">
               18000
            </a>&nbsp;
            JPY
         </font>
         <br>
         <font class="hotellistsmall">
            double
            &nbsp;
            <a href="JavaScript: newWindow=currencyconverter">
               20000
            </a>&nbsp;
            JPY
         </font>
      </td>
   </tr>
</table>
&nbsp;
… □
```

In HTML, the tags (labels) are structure-based rather than content-based. As a consequence, in writing a path to access particular items of data in HTML, we are forced to make certain assumptions about the order of fields as well as the data that we will find in those fields (Firat, Madnick, and Siegel 2000; Mendelzon, Mihaila, and Milo 1996).

We have continued to refer to www.hotelguide.com as a source, but in truth, the problem of identifying a source on the Web becomes much more complex than a relation name. URLs are clearly inadequate because of the temporal nature of data on the Web. Sites hosting dynamic content such as news or financial information are constantly changing. Even a URL with a path expression that specifies order may not suffice to concretely specify a distinct value or its associated attribution path. If we reference a granule by a path, does the path similarly name a source? In this case, a named source can contain a second source perhaps presenting a refined case of composition. (Buneman et al. 1997) has studied aspects of this problem in the context of keys for semistructured data, but the continual challenge will be to extend conclusions to the ad-hoc Web.

Other such pragmatic problems related to the ad-hoc nature of the Web and naming have to do with duplicate sites and whether replicas or mirrors are treated as distinct sources or the same source. Versioning and the temporal nature of the Web in general will also pose problems for attribution.

The Web today almost certainly foreshadows the future of data management. If nothing else, the metaphors carried from the print and publishing world onto the Web will continue in some form tomorrow. Meanwhile, as the Web continues to expand, incorporating ever more data, so to does the need for attribution as a mechanism for managing that growth, whether for search, intellectual property, or evaluating quality. For this reason, extending formal models of attribution (Cui, Widom, and Wiener 1997 (revised 1999); Motro 1996; Rosenthal and Sciore 1999; Sadri 1991; Wang and Madnick 1990) into the semistructured environment is the logical direction to look. The work by Buneman et al. is a terrific start (Buneman, Deutsch, and Tan 1998; Buneman, Khanna, and Tan 2000; 2001; Buneman, Tajima, and Tan 2001).

7 Conclusion

In this thesis excerpt, we explored technologies for addressing attribution in the context of data integration. While data integration is not new, modern information technologies in general and the World Wide Web in particular have made data integration an everyday phenomenon. Web portals, comparison sites, personalized pages, and other examples of on-line integration exacerbate tensions about data quality, intellectual property, and data organization.

In this thesis excerpt, we focus on a technology-oriented approach to the questions of *what* and *where*. We first present a formal model of attribution that represents *what* as a query result and *where* as query inputs. Although our initial interest was sparked by data integration on the Web, we construct our model in terms of the well-understood, logical foundations of the domain relational calculus. Then, beginning with conjunctive queries, we define and evaluate properties of attribution for several different classes of queries. We consider conjunctive queries, conjunctive queries with θ-comparisons (excluding explicit equality), add explicit equality, add union, and finally add negation.

While the domain relational calculus offers a useful framework for developing our model, the definitions are not easily implemented. Consequently, we present an extended relational algebra for attribution. The extended algebra manages attribution in an inductive fashion. Metadata for specifying comprehensive, source, and relevant attribution is associated with every value in a relation; the metadata is updated and carried forward with every successive query operation. After showing some properties relating the extended algebra to the standard relational algebra, we verify that the attribution returned by the algebra corresponds to the attribution defined by the formal model for the same query.

Although our initial interest in attribution stems from the phenomenon of data integration that pervades the Web, we develop our model of attribution in the simpler but more well-understood framework of the relational data model. We conclude Part one of this thesis by returning to the semistructured data models that underlie the Web. We specify some general principles of semistructured data representation and manipulation and then discuss how our attribution intuitions might map onto this semistructured framework.

7.1 Contributions

As noted in Section 2, the problem of attribution has been addressed from a technology perspective as well as a policy perspective many times before. Some of the prior work has focused on domain specific applications (e.g. geographic information systems (Lanter 1991; Woodruff and Stonebraker 1997)) and others have focused on general models. More recently, Buneman et al. (2001) has even developed a formal model for attribution in a semistructured framework. However, we feel that ours is the first to present the problem in a single framework, the dimensions of the attribution problem space, that articulates the relationship between different technology and policy approaches. In addition, we believe that this thesis does provide a number of contributions to both the existing technology literature and the existing policy literature.

The formal model defines three different attribution *types*. *Comprehensive* attribution refers to all query inputs. *Source* attribution refers to the specific inputs in which a specific value appears.[39] *Relevant* attribution asks which query inputs are used to define constraints or restriction conditions on a value of the query result.

We define several properties of attribution and provide a comprehensive analysis of these properties, covering each type of attribution for the full range of relationally complete expressions. We show that strict equivalence for source attribution breaks down under strict equality and that strict equivalence breaks down for all types of attribution upon introduction of union. Attribution composition is particularly useful because it demonstrates that attribution can be constructed inductively and carried forward with the query processing as well as drilled backwards in a step-wise fashion. We show that composition holds for all classes of queries through limited forms of negation and characterize those limited forms of negation.

Recognizing that we might wish to specify results or sources with varying degrees of precision, we introduce the notion of granularity to attribution. Granularity leverages the equivalence property of composition. Attribution is defined for a query result; result granularity attributes a specific subset of values in a result (*what* data is taken) by attributing a composed query that selects the desired values from the initial query result. Because a result granule can itself serve as a source for a composed query, we note the parallel concept of source granularity for specifying a subset of source values (*where* the data comes from).

While ours is not the first extended algebra to address attribution (Motro 1996; Sadri 1991; Wang and Madnick 1990), we prove a number of properties that are left unspoken in earlier work. Relating the extended algebra to the standard relational algebra, we prove that the extended algebra is closed and that it reduces to the standard algebra.

More significantly, our development of a formal model allows us to prove a number of properties not declared in other models. By defining attribution independently of the algebraic definition, we can show that the attribution algebra is a consistent extension of the standard algebra (Dey, Barron, and Storey 1996; Dey and Sarkar 1996). Finally, rather than defining the extended algebra and then simply stating that attribution is defined as whatever the algebra returns, we show that the algebraic attribution for a value in a query result corresponds to the formal model. The formal model allows us to express what is meant by attribution as well as some of its properties. The algebra provides a direct implementation of that model.

[39] For any given value (*what*) in a query result, the source attribution identifies the specific query inputs (*where*) from which the value in the result is drawn.

7.2 Limitations and future work

While this thesis has attempted to cover a great deal of ground, it has also made a number of assumptions and left many issues un-addressed. In this final section, we consider opportunities for future work.

In terms of the formal model, there are a number of opportunities for further work within the existing model and for expanding the current model. In this thesis, granularity is mentioned only as an observation. We need to define granularity formally and define the relationship between attribution for the same query at different levels of granularity. Just as we speculate on converting between different granules, we might speculate on converting between different types of attribution. Finally, attribution refers to the substitutions for unique instances of values in tuples. Therefore, we should consider the role of functional dependencies.

We might also extend the model in at least two directions. First, we should consider whether the formal model can embrace a richer class of queries. The work on data lineage, for example, has extended to aggregation functions and more general classes of functions (Cui, Widom, and Wiener 1997 (revised 1999)). Second, we would like to consider parameterizing the model. Perhaps we could insert specific quality metrics or other measures that are a function of data attribution. (Rosenthal and Sciore 1999), for example, speak of the access constraints on integrated query results.

In considering the algebra, we first must find an algebraic definition for *relevant* variables that corresponds to the formal model. We found syntactic rules that captured a superset of the relevant variables, but had trouble defining a simple function that would support the formal definition.

We would also like to consider the eager, algebraic manipulation of attribution to manage aggregations or other more general classes of functions. Note that the Stanford work addresses the problem in a lazy manner. In addition, we could consider parameterizing the algebra to manage attribution-related metrics. Moreover, we might wish to explore whether the extended algebra is appropriate for managing other types of metadata such as that used experimental data collection (e.g. experimental apparatus, conditions under which the data was collected, etc.)

Extending the formal model to a semistructured data representation is also needed. As noted in Section 6, there are a host of considerations. Naming is a problem. As we commented earlier, how do we reference a source given that URLs are inadequate?[40] A second problem is the management of query composition and granularity. How do we frame these issues in an environment that allows graph restructuring?[41]

While the current thesis focuses on the theory, there is a great deal of opportunity in implementation. An initial algebraic prototype is described in (Lee, Bressan, and Madnick

[40] In Section 6, we observed that the temporal nature of data on the Web as well as dynamic Web sites and personalization (e.g. Web site as modified based upon cookies) can all affect the content referenced by a URL).

[41] As noted in Section 6, the formal model leverages the value-oriented characteristic of the relational data model. In semistructured data, however, different paths (i.e. different structure) can return the same values from the same domains.

1997). In the prototype, attribution is calculated in an eager manner and carried forward with every value. We would also like to implement the algorithm for attribution composition and explore attribution composition as a hybrid lazy-eager approach. Only one step of the attribution is calculated and propagated while enabling a step-wise backwards trace. In the context of the Web, we might consider an attribution Web service to support attribution tracing between integrated query results.

References

2000. Frequently Asked Questions. bookfinder.com, http://www.bookfinder.com/help/faq/.

Aber, Robert E. 1998. H.R. 2652 Testimony on behalf of Information Industry Association. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 12 February. http://www.house.gov/judiciary/41143.htm.

Abiteboul, S. 1997. Querying semistructured data. *International Conference on Database Theory (ICDT `97)*, 8-10 January, in Delphi, Greece.

Abiteboul, Serge, Peter Buneman, and Dan Suciu. 2000. *Data on the Web: From Relations to Semistructured Data and XML*. San Francisco, CA: Morgan Kaufmann Publishers.

Abiteboul, Serge, Ricard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Menlo Park: Addison-Wesley Publishing Company.

Abiteboul, S., D. Quass, J. McHugh, J. Widom, and J. Wiener. 1997. The Lorel query language for semistructured data. *International Journal on Digital Libraries* 1 (1):68-88, April.

Abiteboul, S., and V. Vianu. 1997. Querying the Web. *International Conference on Database Theory (ICDT `97)*, 8-10 January, in Delphi, Greece.

Akamai, White Paper. 2001. Turbo-Charging Dynamic Web Sites with Akamai EdgeSuite. Akamai Technologies, Inc., AKAMWP-TCD1201, http://www.akamai.com/en/resources/pdf/Turbocharging_WP.pdf.

Anderson, William C. 1893. *A Dictionary of Law 1893: A Dictionary and Compendium of American and English Jurisprudence*. Ecclesiastic Commonwealth Community, 2 November 2001 [cited 26 January 2002]. http://ecclesia.org/lawgiver/C.asp.

ASCAP. 2001. *About ASCAP: What Is ASCAP* [cited 20 August 2001 2001]. http://www.ascap.com/about/whatis.html.

Bailey, Joseph P. 1998. Intermediation and electronic markets: Aggregation and pricing in Internet commerce. PhD, Technology, Management and Policy, Massachusetts Institute of Technology, Cambridge.

Baird, Douglas G. 1983. Common Law Intellectual Property and the Legacy of International News Service v. Associated Press. *University of Chicago Law Review* 50:411, Spring.

Band, Jonathan. 1998. *The Digital Millennium Copyright Act, analysis* [Web]. Morrision & Foerster, LLP, Washington, D.C., 20 October 1998 [cited June 2000 2000]. http://www.arl.org/info/frn/copy/band.html.

Band, Jonathan. 1998. Testimony on behalf of the Online Banking Association. Before *Subcommittee on Courts, Intellectual Property and the Administration of Justice*, U.S. House of Representatives. 12 February 1998. http://www.house.gov/judiciary/41148.htm.

Band, Jonathan, and Jonathan S. Gowdy. 1997. Sui generis database protection: has its time come? *D-Lib Magazine*, June, http://www.dlib.org/dlib/june97/06band.html.

Bang, Grace. 1997. European Union Protection of Databases: An Overview of the Database Directive. SUNY Buffalo, http://wings.buffalo.edu/Complaw/CompLawPapers/bang.htm.

Baumol, William, and J. Gregory Sidak. 1994. *Toward competition in local telephony*. AEI studies in telecommunications deregulation, *AEI studies in telecommunications deregulation*. Washington, D.C.: American Enterprise Institute for Public Policy Research.

Berkman, H. 1999. Congress Tackles Database Law. *The National Law Journal*, 22 July.

Bernstein, Philip A., and Thomas Bergstraesser. 1999. Meta-data support for data transformations using Microsoft Repository. *IEEE Data Engineering* 22 (1):9-14.

Besen, Stanley M., Sheila N. Kirby, and Steven C. Salop. 1992. An Economic Analysis of Copyright Collectives. *Virginia Law Review* 78:383, February.

Bloomberg, Michael. 1996. *Michael Bloomberg on WIPO database treaty* [Web news posting] [cited 30 November 2000 1996]. http://www.ainfos.ca/A-Infos96/8/0270.html.

Bloomberg News, Staff. 2001. Bidder's Edge Settle Suits on Web Access. *Los Angeles Times*, 2 March, Sec C, p 2.

BMI. 2001. *BMI Backgrounder* [cited 20 August 2001 2001]. http://www.bmi.com/about/backgrounder.asp.

Bohlen, Michael H., Richard T. Snodgrass, and Michael D. Soo. 1996. Coalescing in Temporal Databases. *Twenty-second International Conference on Very Large Data Bases*, 3-6 September, in Bombay, India, pp 180-91.

Bond, Robert. 1996. *European Union Database Law and the Information Society*. Hobson Audley Hopkins & Wood, 1996 [cited 2 July 2000]. http://ds.dial.pipex.com/town/close/gbb67/itlaw/databas.htm.

Borzo, Jeanette. 2001. Searching: Out of order? *Wall Street Journal*, 24 September, Sec E-Commerce (A Special Report), p R13.

Bray, Tim, Jean Paoli, and C. M. Sperberg-McQueen. 1997. Extensible Markup Language (XML). *XML Journal* 2 (4), Fall.

Bressan, S, C Goh, N Levina, A Shah, S Madnick, and M Siegel. 2000. Context Knowledge Representation and Reasoning in the Context Interchange System. *International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies* 12 (2):165-79, September.

Brown, Mark. 2000. *Zagat Survey: 2000/2001 Philadelphia Restaurants* Edited by M. Klein and N. Gottlieb. New York, NY: Zagat Survey, LLC.

Bulfinch, Thomas. 2001. *Bulfinch's Mythology*. Fisher, Bob, April 2001 [cited 26 January 2002 2002]. http://www.webcom.com/shownet/bulfinch/fables/bull20.html.

Buneman, Peter. 1997. Semistructured data. *Sixteenth ACM Symposium on Principles of Database Systems (PODS)*, 13-15 May, in Tucson, AZ.

Buneman, Peter. 2001. Deep linking (unpublished). University of Pennsylvania.

Buneman, P., S. Davidson, M. Fernandez, and D. Suciu. 1997. Adding structure to unstructured data. *International Conference on Database Theory (ICDT `97)*, 8-10 January, in Delphi, Greece.

Buneman, Peter, Alin Deutsch, and Wang-Chiew Tan. 1998. A deterministic model for semistructured data. *Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats*, http://db.cis.upenn.edu/DL/icdt.ps.gz.

Buneman, Peter, Sanjeev Khanna, and Wang-Chiew Tan. 2000. Data Provenance: Some Basic Issues. *Foundations of Software Technology and Theoretical Computer Science*, 13-15 December, in New Delhi, India.

Buneman, Peter, Sanjeev Khanna, and Wang-Chiew Tan. 2001. Why and Where: A Characterization of Data Provenance. *International Conference on Database Theory (ICDT `01)*, 4-6 January, in London, England, http://db.cis.upenn.edu/DL/whywhere.ps.

Buneman, Peter, Keishi Tajima, and Wang-Chiew Tan. 2001. Deep Citation and Efficient Archiving in Digital Libraries. University of Pennsylvania for Digital Libraries Initiatives II Meeting, http://db.cis.upenn.edu/DL/DL-roanoke.pdf.

CADP, Coalition Against Database Piracy. 2000. *H.R. 354: A Balanced Approach* [cited 2 July 2000]. http://www.gooddata.org/quotes.htm.

Calabresi, Guido, and A. Douglas Melamed. 1972. Property Rules, Liability Rules, and

Inalienability: One View of the Cathedral. *Harvard Law Review* 85 (6):1089, April, http://heinonline.org.

Casey, Tim. 1998. H.R. 2652 Testimony on behalf of the Information Technology Association of America. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 12 February. http://www.house.gov/judiciary/41143.htm.

Chakrabarti, Soumen, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Kleinberg. 1998. Automatic resource compilation by analyzing hyperlink structure and associated text. *Proc. 7th International World Wide Web Conference*, 14-18 April 1998, in Brisbane, Australia, http://decweb.ethz.ch/WWW7/1898/com1898.htm.

Chamberlin, Don, James Clark, Daniela Florescu, Jonathan Robie, Jerome Simeon, and Mugur Stefanescu. 2001. *XQuery 1.0. An XML Query Language*. World Wide Web Consortium, 20 December 2001 [cited 7 July 2001 2001]. http://www.w3.org/TR/2001/WD-xquery-20010607/.

Chamberlin, Don, Peter Fankhauser, Massimo Marchiori, and Jonathan Robie. 2001. *XML Query Use Cases: W3C Working Draft 08 June 2001*. World Wide Web Consortium, 20 December 2001 [cited 17 August 2001]. http://www.w3.org/TR/xmlquery-use-cases.

Chawathe, S., H. Garca-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. 1994. The TSIMMIS project: Integration of heterogeneous information sources. *Information Processing Scoiety of Japan*, October, in Tokyo, Japan.

Chawathe, Sudarshan S., Serge Abiteboul, and Jennifer Widom. 1999. Managing Historical Semistructured Data. *Theory and Practice of Object Systems* 24 (4):1, 1999.

Clark, James, and Steve DeRose. 2001. *XML Path Language (XPath) Version 1.0: W3C Recommendation 16 November 1999* [cited 17 August 2001]. http://www.w3c.org/TR/1999/REC-xpath-19991116.

Coase, Ronald. 1988. The Nature of the Firm (1937). In *The Firm, the Market, and the Law*. Chicago, IL: University of Chicago Press.

Cohen, Julie E. 1997. Some reflections on copyright management systems and laws designed to protect them. *Berkeley Technology Law Journal* 12 (1), http://www.law.berkeley.edu/journals/btlj/articles/12_1/Cohen/html/text.html.

Constant, Beth A. 2000. Chalk Talk: The Fair Use Doctrine: Just What Is Fair? *Journal of Law and Education* 29:385, July.

Corlin, Richard F. 1998. H.R. 2652 Statement of the American Medical Association. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 12 February. http://www.house.gov/judiciary/41146.htm.

Cronk, Denis R. 2000. *Tighter Protection Against Piracy of Online Data: Top NAR Legislative Priority*. National Association of Realtors, 6 January 2000 [cited 30 November 2000]. http://nar.realtor.com/news/2000Releases/January/6.htm.

Cui, Claire Yingwei, and Jennifer Widom. 2000. Practical Lineage Tracing in Data Warehouses. *International Conference on Data Engineering*, February, in San Diego, California, http://www-db.stanford.edu/pub/papers/trace.ps.

Cui, Claire Yingwei, and Jennifer Widom. 2001. Lineage Tracing for General Data Warehouse Transformations. *27th International Conference on Very Large Data Bases (VLDB)*, 11-14 September, in Rome, Italy, http://dbpubs.stanford.edu:8090/pub/2001-5.

Cui, Claire Yingwei, Jennifer Widom, and Janet L. Wiener. 1997 (revised 1999). Tracing the Lineage of View Data in a Datawarehousing Environment. Stanford University, http://www-db.stanford.edu/pub/papers/lineage-full.ps.

databasedata.org. 1999. A Basic Guide to Database Legislation in the 106th Congress. databasedata.org, http://www.databasedata.org/db101/db101.html.

databasedata.org. 1999. Side-By-Side Comparison of Database Protection Bills. databasedata.org, http://www.databasedata.org/DBside-by-side.

deBakker, Bas, and Irsan Widarto. 2001. *An Introduction to XQuery*. X-Hive Corporation, 13 December [cited 13 December 2001]. http://www.perfectxml.com/articles/xml/xquery.asp.

Desai, B. C., P. Goya, and F. Sadri. 1987. Non-first normal form universal relations: an application to information retrieval systems. *Information Systems* 12 (1):49-55, 1987.

Dey, Debabrata, Terence Barron, M., and Veda C. Storey. 1996. A complete temporal relational algebra. *VLDB Journal* 5:167-180.

Dey, Debabrata, and Sumit Sarkar. 1996. A Probabilistic Relational Model and Algebra. *ACM Transactions on Database Systems* 21 (3):339-369, September.

Djavaherian, David. 1998. Hot News and No Cold Facts: NBA v. Motorola and the Protection of Database Contents. *Richmond Journal of Law and Technology* 5 (2), Winter, http://www.richmond.edu/~jolt/v5i2/djava.html.

DOS, Department of State Bureau of Administration. 2001. *Key Officers List, Japan*. U.S. Department of State, 18 October 2001 [cited 2001]. http://www.foia.state.gov/mms/KOH/keypostdetails.asp?post=0&letter=J&id=75.

Drahos, Peter. 1996. *A philosophy of intellectual property*. Brookfield, Vermont: Dartmouth Publishing Company.

Duncan, Daniel C. 1999. H.R. 354 Testimony on behalf of the Software and Information Industry Association. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 18 March 1999. http://www.house.gov/judiciary/106-dunc.htm.

Duschka, Oliver, and Michael Genesereth. 1997. Answering recursive queries using views. *Sixteenth ACM Symposium on Principles of Database Systems (PODS)*, 13-15 May, in Tucson, AZ.

Duschka, Oliver M. , and Michael R. Genesereth. 1997. Query Planning in Infomaster. *ACM Symposium on Applied Computing*, February, in San Jose, CA.

eBay. 2000. *eBay, Inc.v. Bidder's Edge Inc.*, U.S. District Court for the Northern District of California:LEXIS 13326 (21 July).

Effross, Walter A. 1998. Withdrawl of the reference: rights, rules, and remedies for unwelcomed Web-linking. *South Carolina Law Review* 49:651-593, http://www.wcl.american.edu/pub/faculty/effross/withdrawl.html.

Elgison, Martin, and James M. Jordan. 1997. Trademark cases arise from meta-tags, frames: disputes involve search-engine indexes, web sites within web sites, as well as hyperlinking. *National Law Journal*, 20 October, http://cyber.law.harvard.edu/metaschool/fisher/linking/framing/mixed1.html.

Elkin-Koren, Niva. 1997. Copyright policy and the limits of freedom of contract. *Berkeley Technology Law Journal* 12 (1), http://www.law.berkeley.edu/journals/btlj/articles/12-1/koren.html.

Feist v. Rural. 1991. *Feist Publications, Inc. v. Rural Telephone Service*, U. S. Supreme Court 499:340 (1991).

Ferber, Don. 1991. Tracking Tiger: The use, verifcation, and updating of tiger data. *GIS/LIS*, 1991, in Atlanta, Georgia, pp 230-239.

Ferber, Don. 1992. GIS project documentation: The Wisconsin TIGER Project example. *GIS/LIS (1992)*, 10-12 November, in San Jose, California, pp 221-230.

Fernandez, M., D. Florescu, J. Kang, A. Levy, and D. Suciu. 1997. Strudel: A web site management system. *ACM SIGMOD Conference on Management of Data*, 13-15 May, in Tucson, AZ.

Fernandez, M., D. Florescu, A. Levy, and D. Suciu. 1997. A query language for a web-site management system. *SIGMOD Record* 26 (3):4-11, September.

Fernandez, Mary, and Jonathan Marsh. 2001. *XQuery 1.0 and XPath 2.0 Data Model: W3C Working Draft 7 June 2001*. World Wide Web Consortium, 20 December [cited 7 July 2001]. http://www.w3.org/TR/2001/WD-query-datamodel-20010607/.

Fernandez, Mary, and Jonathan Robie. 2001. *XML Query Data Model: W3C Working Draft 11 May 2000*. World Wide Web Consortium [cited 7 July 2001]. http://www.w3.org/TR/2000/WD-query-datamodel-20000511.

Ferri, Lisa M., and Robert G. Gibbons. 2000. Forgive Us Our Virtual Trespasses: The 'eBay' Ruling. *New York Law Journal*:1, 27 June 2000.

Firat, A., S. Madnick, and M. Siegel. 2000. The Cameleon Web Wrapper Engine. *VLDB Workshop on Technologies for E-Services*, in Cairo, Egypt.

Florescu, Daniela, Alon Y. Levy, and Alberto Mendelzon. 1998. Databse Techniques for the World-Wide-Web: A Survey. *SIGMOD Record* 1998.

Fry, Jason. 2001. Why Shopper's Loyalty To Familiar Web Sites Isn't So Crazy After All. *Wall Street Journal*, 13 August, Sec Marketplace, p B1.

Fujita, Anne K. 1996. The Great Internet Panic: How Digitization is Deforming Copyright Law. *Journal of Technology Law & Policy* 2 (1), http://journal.law.ufl.edu/~techlaw/2/fall96index.html.

Gale Research, Inc. 1999. *Gale Directory of Databases*. Detroit, MI: Gale Research, Inc.

Garland, Susan. 1999. Whose Info Is It Anyway? *Business Week*, 13 September, 114.

Gibbons, Robert. 1992. *Game Theory for Applied Economists*. Princeton, NJ: Princeton University Press.

Ginsburg, Jane C. 1990. Creation and Commercial Value: Copyright Protection of Works of Information. *Columbia Law Review* 90:1865, November.

Ginsburg, Jane C. 1992. No "Sweat"? Copyright and Other Protection of Works of Information

After Feist v. Rural Telephone. *Columbia Law Review* 92:338, March.

Ginsburg, Jane C. 1997. Statement on H.R. 2652: The Collections of Information Antipiracy Act. Before *Subcommittee on Courts, Intellectual Property and the Administration of Justice*, U.S. House of Representatives. 28 October 1997. http://www.house.gov/judiciary/41147.htm.

Goh, Cheng Hian. 1997. Representing and reasoning about semantic conflicts in heterogeneous information systems. Doctor of Philosophy, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Goh, Cheng Hian, S Bressan, S Madnick, and M Siegel. 1999. Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. *ACM Transactions on Office Information Systems*, July.

Goldstein, Paul. 1994. Toward a Third Intellectual Property Paradigm: Comments: Comments on a Manifesto Concerning the Legal Protection of Computer Programs. *Columbia Law Review* 94:2573, December.

Gordon, Wendy J. 1992. On Owning Information: Intellectual Property and the Restitutionary Impulse. *Virginia Law Review* 78:149, February.

Gordon, Wendy J. 1992. Asymmetric Market Failure and Prisoner's Dilemma in Intellectual Property. *University of Dayton Law Review* 17:853, Spring.

Gordon, Wendy J. 1994. Toward a Third Intellectual Property Paradigm: Comments: Assertive Modesty: An Economics of Intangibles. *Columbia Law Review* 94:2579, December.

Gorman, Robert A., and Jane C. Ginsburg. 1993. *Copyright for the Nineties*. Fourth ed. Charlottesville, VA: Michie Company.

Grady, Richard K. 1988. Data lineage in land and geographic information systems (LIS/GIS). *GIS/LIS (88)*, 30 November - 2 December, in San Antonio, Texas.

Green, Robert. 2000. *eBay Revisited*. the Synthesis, 1 July [cited 3 December 2000]. http://www.synthesis.net/columns/websight/07/01.

Grimm, Brothers. 2000. *Grimm's Fairy Tales "Hansel and Gretel"* [Web]. Mordent Software [cited 30 June 2000 2000]. http://www.mordent.com/folktales/grimms/hng/hng.html.

Grimm, Brothers, Josef Scharl Scharl, Jacob Ludwig Carl Grimm, and Wilhelm Grimm. 1976. *The Complete Grimm's Fairy Tales (Pantheon Fairy Tale and Folklore Library)* Edited by

J. Stern: Random House.

Grosso, Paul, and Norman Walsh. 2000. XSL Concepts and Practical Use. *XML Europe 2000*, 12 June, in Paris, France, http://www.nwalsh.com/docs/tutorials/xsl/xsl/slides.html.

Guelich, Scott, Shishir Gundavaram, and Gunther Birznieks. 2000. *CGI Programming with Perl*. 2nd ed. Sebastopol, CA: O'Reilly & Associates, Inc.

H.R. 354. 1999. *Collections of Information Antipiracy Act*. R. H. Coble:To amend title 17, United States Code, to provide protection for certain collections of information., U.S. House of Representatives, 106th Congress, 19 January.

H.R.1858. 1999. *Consumer and Investor Access to Information Act of 1999*. R. T. Bliley:To promote electronic commerce through improved access for consumers to electronic databases, including securities market information databases., U.S. House of Representatives, 106th Congress, 19 May 1999.

H.R. 2652. 1997. *Collections of Information Antipiracy Act*. R. H. Coble:To amend title 17, United States Code, to prevent the misappropriation of collections of information., U.S. House of Representatives, 105th Congress, 9 October.

H.R. 3531. 1996. *Database Investment and Intellectual Property Antipiracy Act of 1996*. R. C. J. Moorehead:To amend title 15, United States Code, to promote investment and prevent intellectual property piracy with respect to databases., U.S. House of Representatives, 104th Congress, 23 May.

Hammack, William. 1998. H.R. 2652 Testimony on behalf of the Association of Directory Publishers. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 12 February. http://www.house.gov/judiciary/41146.htm.

Hardy, Trotter. 1995. Contracts, Copyright, and Premeption in a Digital World. *Richmond Journal of Law and Technology* 1 (2), http://www.urich.edu/olt/v1i1/hardy.html.

Hardy, Trotter. 1996. Property (and Copyright) in Cyberspace. *The University of Chicago Legal Forum*:217.

Hawkins, Jennifer L. 1997. ProCD, Inc. v. Zeidenberg: Enforceability of shrinkwrap licenses under the Copyright Act. *Richmond Journal of Law and Technology* 3 (1), http://www.richmond.edu/~jolt/v3il/hawkins.html.

Henderson, Lynn O. 1999. H.R. 354 Testimony on behalf of Agricultural Publisher's Association. Before *Subcommittee on Courts and Intellectual Property of the Committee on* *the Judiciary*, US House of Representatives, Washington, DC. 18 March 1999. http://www.house.gov/judiciary/106-hend.htm.

Hitchcock, Steve, L. Carr, S. Harris, J. Hey, and W. Hall. 1997. Citation linking: Improving access to online journals. *2nd ACM International Conference on Digital Libraries*, 23-26 July, in Philadelphia, PA, pp 115-122, http://journals.ecs.soton.ac.uk/acmdl97.htm.

Horbaczewski, Henry. 1999. On behalf of the Coalition Against Database Piracy on H.R. 1858, the Consumer and Investor Accessto Information Act of 1999. Before *Subcommittee on Telecommunications, Trade and Consumer Protection of the House Commerce Committee*, US House of Representatives, Washington, DC. 15 June 1999. http://www.gooddata.org/Horbaczewski_testimony.htm.

hotelguide.com. 2001. *Hotelguide.com - Book your accomodation online from our International Hotel Directory*. hotelguide.com [cited 20 August 2001]. http://www.hotelguide.com.

Howe, Dennis, ed. 2000. *Free On-line Dictionary of Computing*: Imperial College Department of Computing.

Hu, Jim. 2000. *MP3.com pays $53.4 million to end copyright suit*. CNET News.com, 15 November, 11:20 am PT [cited 3 December 2000]. http://news.cnet.com/news/0-1005-202-3681102.html.

Huang, Kuan-Tsae, Yang W. Lee, and Richard Y. Wang. 1999. *Quality Information and Knowledge*. Upper Saddle River, NJ: Prentice Hall PTR.

Hunsucker, G.M. 1997. The European Database Directive: Regional stepping stone to an international model? *Fordham Intellectual Property, Media and Entertainment Law Journal* 7.

IFLA, International Federation of Library Associations. 2002. *Committee on Copyright and other Legal Matters*. IFLA, 22 November 2001 [cited September 2001]. http://www.ifla.org/III/clm/copyr.htm.

INS v. AP. 1918. *International News Service v. Associated Press*, U.S. Supreme Court 248:215 (1918).

Japan Youth Hostels, Inc. 2001. *Tokyo, Japan Youth Hostels*. Hostelling International [cited 20 August 2001]. http://www.jyh.or.jp/olhb/JYH-English/jyh.kantou/jyh-7.13.html.

Junnarkar, Sandeep. 1999. Ticketmaster Online-CitySearch buys Sidewalk. *CNET News.com*, 19 July 1999, 12:20 PT, http://www.canada.cnet.com/news/0-1005-200-345004.html.

Kaplan, Carl S. 1999. A search site for search sites is accused of trespassing. *New York Times*, 24 September 1999, http://www.nytimes.com/library/tech/99/09/cyber/cyberlaw/24law.html.

Kaplan, Carl S. 2000. Judge says a spider is trespassing on eBay. *New York Times*, 26 May, http://www.nytimes.com/library/tech/00/05/cyber/cyberlaw/26law.html.

Karjala, Dennis S. 1994. Toward a Third Intellectual Property Paradigm: Comments: Misappropriation as a Third Intellectual Property Paradigm. *Columbia Law Review* 94:2594, December.

Katz, Howard. 2001. *An introduction to XQuery*. IBM developer works XML zone articles, June [cited 13 December 2001]. http://www-106.ibm.com/developerworks/xml/library/x-xqury.html.

Kinko's. 1991. *Basic Books, Inc., Harper & Row Publishers, Inc., John Wiley & Sons, Inc., McGraw-Hill, Inc., Penguin Books USA, Inc, Prentice-Hall, Inc., Richard D. Irwin, Inc., and William Morrow & Co., Inc., v. Kinko's Graphics Corporation*, United States District Court for the Southern District of New York 758:1522 (28 March).

Kirkman, Catherine Sansum. 1998. *Legal Protection of Online Databases*. WebTechniques [cited 2 July 2000]. http://www.webtechniques.com/archives/1998/01/just/.

Kleinberg, Jon. 1998. Authoritative sources in a hyper-linked environment. *Proceedings, 9th ACM-SIAM Symposium on Discrete Algorithms*, http://www.cs.cornell.edu/home/kleinber/auth.pdf.

Klug, A. 1988. On Conjunctive Queries Containing Inequalities. *Journal of the Association for Computing Machinery* 35 (1):146-160.

Konopnicki, D., and O. Shmueli. 1995. W3QS: A query system for the World Wide Web. *21st International Conference on Very Large Data Bases (VLDB)*, 11-15 September, in Zurich, Switzerland, pp 54-65.

Kravitz, Mark. 2001. *$18 and Under: The Guide to Reasonable Dining and Entertainment*. Third ed. Philadelphia, PA: Spirit of `76 Publishing.

Krebs, Brian. 2000. Law profs oppose Court's ban on eBay spidering. *eMarketer*, 3 December, http://www.emarketer.com/enews/20000719_spidering.html.

Krummenacker, Markus. 1995. *Are "Intellectual Property Rights" Justified?* [Web] [cited 11 July 2001].

Kuester, Jeffrey R., and Peter A. Nieves. 1997. What's all the hype about hyperlinking? Thomas, Kayden, Horstemeyer & Risley, L.L.P., http://www.tkhr.com/articles/hyper.html.

Langin, Dan, and James Cary Howell. 2000. ISP Risk Management. *Boardwatch Magazine*, August, 82-6.

Lanter, David P. 1991. Design of a lineage-based meta-data base for GIS. *Cartography and Geographic Information Systems* 18 (4):255-261.

Lanter, David P., and Chris Surbey. 1994. Metadata analysis of GIS data processing, a case study. *International Symposium on Spatial Data Handling (6th)*, 1994, in Edinburgh, Scotland, pp 314-324.

Lasswell, Harold. 1948. The Structure and Function of Communication in Society. In *The Communication of Ideas, A Series of Addresses*, edited by L. Bryson. New York, NY: Institute for Religious and Social Studies, distributed by Harper.

Lederberg, Joshua. 1999. H.R. 354 Testimony on behalf of the National Academy of Sciences, National Academy of Engineering, Institute of Medicine, and the American Association for the Advancement of Science. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 18 March 1999. http://www.house.gov/judiciary/106-pinc.htm.

Lee, T., and S. Bressan. 1997. Multimodal Integration of Disparate Information Sources with Attribution. *ER 97 Workshop on Information Retrieval and Conceptual Modeling*, November, in Los Angeles, CA.

Lee, T., S. Bressan, and S. Madnick. 1997. Source Attribution for Querying Against Semi-structured Documents. MIT Sloan School of Management, Sloan WP#4042 CISL WP#99-01.

Lee, T., S. Bressan, and S. Madnick. 1998. Source Attribution for Querying Against Semi-stuctured Documents. *Workshop on Web Information and Data Management, Seventh International ACM Conference on Information and Knowledge Management*, 3-7 November, in Bethesda, MD.

Lee, T., M. Chams, R. Nado, S. Madnick, and M. Siegel. 1999. Information Integration with Attribution Support for Corporate Profiles. *Eighth International ACM Conference on Information and Knowledge Management (CIKM)*, 2-6 November, in Kansas City, KS, pp 423-430.

Lenz, Evan. 2001. *XQuery: Reinventing the Wheel?* XYZFind Corp. [cited 13 December

2001]. http://xmlportfolio.com/xquery.html.

Let's Go, Inc. 1993. *Let's Go: Germany, Austria & Switzerland* Edited by G. W. Rodkey. New York, NY: St. Martin's Press.

Levy, Alon Y. 2000. Logic-Based Techniques in Data Integration. In *Logic Based Artificial Intelligence*, edited by J. Minker: Kluwer Publishers.

Levy, Alon Y., Anand Rajaraman, and Joann J. Ordille. 1996. Querying Heterogeneous Information Sources Using Source Descriptions. *22nd International Conference on Very Large Data Bases (VLDB)*, 3-6 September, in Bombay, India.

Lindemans, Micha F. 2000. *The Encyclopedia Mythica* [cited 6/30/2000 2000]. http://www.pantheon.org/mythica/areas/greek.

Linn, Anne. 2000. *History of Database Protection: Legal Issues of Concern to the Scientific Community*. National Research Council, 3 March 2000 [cited 2 July 2000]. http://www.codata.org/codata/data_access/linn.html.

Litman, Jessica. 1992. After Feist. *University of Dayton Law Review* 17.

Liu, H. C., and K Ramamohanarao. 1994. Algebraic equivalences among nested relational expressions. The University of Melbourne, Technical Report 94/4, http://http://www.cs.mu.oz.au/publications/tr_db/mu_94_04.ps.gz.

Liu, Joseph P. 2001. Owning digital copies: Copyright law and the incidents of copy ownership. *William and Mary Law Review* 42:1245-1366, April, 2001.

Lutzker, Arnold P. 1999. *Primer on the Digital Millennium*. Lutzker and Lutzker, LLP, Washington, D.C., 5 February 1999 [cited June 2000]. http://www.arl.org/info/frn/copy/primer.html.

MacMillan, Robert. 2000. Sen. DeWine calls for database bill next year. *Newsbytes*, 26 October, 10:06 AM EST, http:////www.newsbytes.com/news/00/157254.html.

Mahoney, Paul G. 1997. Technology, Property Rights in Information, and Securities Regulation. *Washington University Law Quarterly* 75 (2):815, Summer.

Maier, David. 1983. *The theory of relational databases*. Rockville, Maryland: Computer Science Press.

Marino, Fabio. 2000. *Database Protection in the European Union* [cited 2 July 2000]. http://www.jus.unitn.it/cardozo/Review/Students/Marino1.html.

Markon, Jerry. 2001. E-Business: The Web @ Work/Willkie Farr & Gallagher. *Wall Street Journal*, 30 April, Sec E-Business, p B5.

McDermott, Terry. 1999. H.R. 354 Testimony on behalf of National Association of Realtors. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 18 March 1999. http://www.house.gov/judiciary/106-mcde.htm.

McHugh, J., S. Abiteboul, R. Goldman, D. Quass, and J. Widom. 1997. Lore: A database management system for semistructured data. *ACM SIGMOD Record* 26 (3):54-66, 1997.

Mendelzon, Alberto, George A. Mihaila, and Tova Milo. 1996. Querying the World Wide Web. *Fourth International Conference on Parallel and Distributed Information Systems (PDIS)*, 18-20 December, in Miami, FL, pp 80-91.

Mendelzon, Alberto, and Tova Milo. 1997. Formal models of Web queries. *Sixteenth ACM Symposium on Principles of Database Systems (PODS)*, 13-15 May, in Tucson, AZ, pp 134-143.

Merges, Robert P. 1994. Toward a Third Intellectual Property Paradigm: Comments: Of Property Rules, Coase, and Intellectual Property. *Columbia Law Review* 94:2655, December.

Merges, Robert P. 1996. Contracting into Liability Rules: Intellectual Property Rights and Collective Rights Organizations. *California Law Review* 84 (5):1293, October, http://www.sims.berkeley.edu/BCLT/pubs/merges/contract.htm.

Merges, Robert P., Peter S. Menell, Mark A. Lemley, and Thomas M. Jorde. 1997. *Intellectual Property in the New Technological Age*. New York: Aspen Law & Business, Aspen Publishers, Inc.

Mihaila, George A., Louiqa Raschid, and Maria Esther Vidal. 1999. Querying "Quality of data" metadata. *IEEE Metadata*, 1999, http://www.computer.org/conferen/proceed/meta/1999/papers/65/gmihaila.html.

Milgrom, Paul, and John Roberts. 1992. *Economics, Organization and Management*. Englewood Cliffs, NJ: Prentice Hall.

Minker, Jack, ed. 1988. *Foundations of Deductive Databases and Logic Programming*. Los Altos: Morgan Kaufmann Publishers, Inc.

Monster.com. 2000. *Monster.com Joins Coalition Against Database Piracy*, 26 September

2000 [cited 30 November 2000]. http://www.gooddata.org/monster.htm.

Motro, Amihai. 1996. Panorama: a database system that annotates its answers to queries with their properties. *Journal of Intelligent Information Systems* 7 (1):51-73.

Motro, Amihai, and Igor Rakov. 1998. Estimating the quality of databases. *Flexible Query Answering Systems. Third International Conference, FQAS'98. Proceedings*, 13-15 May, in Roskilde, Denmark, pp 298-307.

MP3.com. 2000. *UMG Recordings, Inc. v. MP3.com, Inc.*, United States District Court for the Southern District of New York:LEXIS 13293 (6 September).

Nazareth, Annette L. 1999. Prepared statement on behalf of the Securites and Exchange Commission concerning H.R. 1858. Before *Subcommittee on Finance and Hazardous Materials*, U.S. House of Representatives, Washington, D.C. 30 June 1999.

NBA v. Motorola. 1997. *National Basketball Association v. Motorola, Inc.*, 2nd Cricuit 105:841.

NCID, National Center for Infectious Diseases. 2001. Travelers' Health: Health Information for Travelers to East Asia. U.S. Department of Health and Human Services, Centers for Disease Control (CDC), http://www.cdc.gov/travel/eastasia.htm.

Neal, James G. 1999. H.R. 354 Testimony on behalf of American Association of Law Libraries, American Library Association, Association of Research Libraries, Medical Library Association, and Special Libraries Association. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 18 March. http://www.house.gov/judiciary/106-neal.htm.

Nestorov, S., S. Abietboul, and R. Motwani. 1997. Inferring structure in semistructured data. *Workshop on Management of Semistructured Data in Conjunction with ACM SIGMOD*, 13-15 May, in Tucson, AZ.

NFL v. Delaware. 1977. *National Football League v. State of Delaware*, F. Supp. 435:1372.

Nicolas, Jean-Marie. 1982. Logic for Improving Integrity Checking in Relational Data Bases. *Acta Informatica* 18:227-53.

Nimmer, Raymond T. 1998. Breaking Barriers: The Relation Between Contract and Intellectual Property Law. *Berkeley Technology Law Journal* 13:827, Fall.

Nissen, Dinah, and Jamie Barber. 1996. The EC Database Directive. *In-House Lawyer*, May, http://www.harbottle.co.uk/pubs/may96.htm.

Nottrott, Rudolf W., Matthew B. Jones, and Mark Schildhauer. 1999. Using XML-structured metadata to automate quality assurance processing for ecological data. *IEEE Metadata*, http://www.computer.org/conferen/proceed/meta/1999/papers/64/rnottrott.html.

NRC, National Research Council. 1997. *Bits of Power: Issues in Global Access to Scientific Data*. Computer Science and Telecommunications Board, *Computer Science and Telecommunications Board*. Washington, DC: National Academy Press.

NRC, National Research Council. 1997. *For the Record: Protecting Electronic Health Information*. Computer Science and Telecommunications Board, *Computer Science and Telecommunications Board*. Washington, DC: National Academy Press.

NRC, National Research Council. 1999. *A Question of Balance: Private Rights and Public Interest in Scientific and Technical Databases*. Commission on Physical Sciences, Mathematics, and Applications, *Commission on Physical Sciences, Mathematics, and Applications*. Washington, DC: National Academy Press.

NRC, National Research Council. 2000. *The Digital Dilemma: Intellectual Property in the Information Age*. Engineering and Physical Sciences, *Engineering and Physical Sciences*. Washington, DC: National Academy Press.

Olsen, Stefanie. 1999. *eBay inks deal with auction search site*. CNET News.com, 1 December 1999 2:40 pm PST [cited 3 December 2000]. http://news.cnet.com/news/0-2007-300-1475546.html.

OMM, O'Melveney & Meyers LLP. 1999. *Copyright Law and the Internet*. O'Melveney & Meyers LLP, 19 November 1998 [cited 16 April 1999]. http://www.omm.com/ilpg/ip/copyright.html.

O'Rourke, Maureen A. 1997. Copyright Preemption After the ProCD Case: A Market-Based Approach. *Berkeley Technology Law Journal* 12 (1), http://www.law.berkeley.edu/journals/btlj/articles/12-1/ORourke.html.

O'Rourke, Maureen A. 1999. Progressing Towards a Uniform Commercial Code for Electronic Commerce or Racing Towards Nonuniformity? *Berkeley Technology Law Journal* 14 (2), http://www.law.berkeley.edu/journals/btlj/articles/14_2/O'Rourke/html/reader.html.

OSTP, Office of Science and Technology Policy. 1999. Administration testimony on HR 354. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 18 March 1999. http://www.whitehouse.gov/WH/EOP/OSTP/html/19993_19_2.html.

Paepke, C. Owen. 1987. An Economic Interpretation of the Misappropration Doctrine: Common Law Protection for Investments in Innovation. *High Technology Law Journal*.

Papakonstantinou, Y., S. Abiteboul, and H. Garca-Molina. 1996. Object fusion in mediator systems. *22nd International Conference on Very Large Data Bases (VLDB)*, 3-6 September, in Bombay, India.

Papakonstantinou, Y., H. Garca-Molina, and J. Widom. 1995. Object exchange across heterogeneous information sources. *International Conference on Data Engineering*, in Taipei, Taiwan, pp 251-260.

Patterson, L. Ray. 1992. Copyright Overextended: A Preliminary Inquiry Into the Need for a Federal Statute of Unfair Competition. *Dayton Law Review* 17:385, Winter.

Perritt, Henry H. Jr. 1996. Property and Innovation in the Global Information Infrastructure. *The University of Chicago Legal Forum*:261.

Peters, Marybeth. 1999. H.R. 354 Testimony for the U.S. Copyright Office. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 18 March 1999. http://www.house.gov/judiciary/106-pete.htm.

Phelps, Charles E. 1999. H.R. 354 Testimony on behalf of the Association of American Universities, the American Council of Education, and the National Association of State Universities and Land-Grant Colleges. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 18 March 1999. http://www.house.gov/judiciary/106-phel.htm.

Pincus, Andrew J. 1999. H.R. 354 Testimony for the U. S. Department of Commerce. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 18 March 1999. http://www.house.gov/judiciary/106-pinc.htm.

Pindyck, Robert S., and Daniel L. Rubinfeld. 1992. *Microeconomics*. Second ed. New York, NY: Macmillan Publishing Co.

Planet, Lonely. 2001. *Worldguide, Destination: Tokyo*. Lonely Planet [cited 20 August 2001]. http://www.lonelyplanet.com/destinations/north_east_asia/tokyo/attractions.htm.

Pollack, Malla. 1999. The Right to Know?: Delimiting Database Protection at the Juncture of the Commerce Clause, the Intellectual Property Clause, and the First Amendment. *Cardozo Arts & Entertainment Law Journal* 17.

Posner, Richard A. 1992. *Economic Analysis of Law*. 4th ed. Boston, MA: Little, Brown.

Princeton v. MDS. 1992. *Princeton University Press, Macmillan, Inc. and St. Martin's Press, Inc. v. Michigan Document Services, Inc. and James M. Smith*, U. S. District Court for the Eastern District of Michigan, Southern Division 1992:13257 (2 April).

Princeton v. MDS. 1996. *Princeton University Press, Macmillan, Inc. and St. Martin's Press, Inc. v. Michigan Document Services, Inc. and James M. Smith*, United States Court of Appeals for the Sixth Circuit 99:1381 (8 November).

ProCD v. Zeidenberg. 1996. *ProCD v. Zeidenberg*, 7th Circuit 908:640.

Quass, D., A. Rajaraman, Y Sagiv, J. Ullman, and J. Widom. 1995. Querying semistructured heterogeneous information. *Fourth International Conferenc on Deductive and Object-Oriented Databases*, in Singapore, pp 436-445.

Quass, D., J. Widom, R. Goldman, K. Haas, Q. Luo, J. McHugh, S. Nestorov, A. Rajaraman, H. Rivero, S. Abiteboul, J. Ullman, and J. Wiener. 1996. LORE: A Lightweight Object REpository for semistructured data. *ACM SIGMOD International Conference on Management of Data*, June, in Montreal, Canada.

Raggett, Dave. 2000. *Adding a touch of style*. W3C, 29 August 2000 [cited 23 October 2001]. http://www.w3.org/MarkUp/Guide/Style.

Raggett, Dave. 2001. *Getting started with HTML*. W3C, 4 June 2001 [cited December 2000]. http://www.w3.org/MarkUp/Guide/.

Ramakrishnan, Raghu, and Johannes Gehrke. 2000. *Database Management Systems*. 2nd ed. Boston, MA: McGraw-Hill.

Raskind, Leo J. 1991. The Misappropriation Doctrine as a Competitive Norm of Intellectual Property Law. *Minnesota Law Review* 75:875, February.

Raul, Alan Charles, Edward R. McNicholas, and Claudia A. von Pervieux. 2000. *Who Owns the Data? Evolving Protections for Facts, Secrets and Personal Information in Cyberspace* [Web]. Washington, D.C. Office of Sidley & Austin, April 2000 [cited 11 December 2000]. http://www.sidley.com/cyberlaw/features/protect.asp.

Reichman, J.H., and Pamela Samuelson. 1997. Intellectual property rights in data? *Vanderbilt Law Review* 50, January.

Reichman, J. H., and Paul F. Uhlir. 1999. Database protection at the crossroads: Recent developments and their impact on science and technology. *Berkeley Technology Law*

Journal 14 (2), http://www.law.berkeley.edu/journals/btlj/articles/14_2/Reichman/html/reader.html.

RIAA. 2000. *MP3.com Lawsuit Q&A*. Recording Industry Association of America [cited 3 December 2000]. http://www.riaa.com/MP3lawsuit.cfm.

Rob, Peter, and Carlos Coronel. 1997. *Databsae Systems: Design, Implementation, and Management*. Cambridge, MA: Course Technology, International Thomson Publishing.

Rosenbaum, David E. 2000. Database Legislation Spurs Fierce Lobbying. *New York Times*, 5 June, Sec A, p 14, http://www.gooddata.org/NYT.htm.

Rosenthal, A., and E. Sciore. 1999. Security administration for federations, warehouses, and other derived data. *IFIP WG11.3 Conference on Database Security*, http://www.cs.bc.edu/~sciore/papers/IFIP99.pdf.

Rosenthal, A., and E. Sciore. 1999. Administering propagated metadata in large, multi-layer database systems. *IEEE Workshop on Knowledge and Data Exchange*, 7 November, http://www.cs.bc.edu/~sciore/papers/KDEX99.pdf.

Roth, Mark A., Henry F. Korth, and Abraham Silberschatz. 1988. Extended Algebra and Calculus for Nested Relational Databases. *ACM Transactions on Database Systems* 13 (4):389-417, December.

Rough Guides, Travel. 2001. *Rough Guide Travel: Tokyo*. Rough Guide Travel [cited 20 August 2001 2001]. http://travel.roughguides.com/content/10072/22912.htm.

S. 95. 1999. *Trading Information Act*. S. J. McCain:To amend the Communications Act of 1934 to ensure that public availability of information concerning stocks traded on an established stock exchange continues to be freely and readily available to the public through all media of mass communication., U.S. Senate, 106th Congress, 1st session, 19 January 1999.

S. 2291. 1998. *Collections of Information Antipiracy Act*. S. R. Grams:A bill to amend title 17, United States Code, to prevent the misappropriation of collections of information, U.S. Senate, 105th Congress, 10 July.

Sableman, Mark. 1999. Link Law: The emerging law of Internet hyperlinks. *Communication Law and Policy* 4 (4):557-601, http://www.ldrc.com/cyber2.html.

Sadri, Fereidoon. 1991. Modeling uncertainty in databases. *International Conference on Data Engineering*, 8-12 April, in Kobe, Japan, pp 122-131.

Sadri, Fereidoon. 1994. Aggregate operations in the information source tracking method. *Theoretical Computer Science* 133 (2):421-442, 24 October.

Sadri, Fereidoon. 1995. Information source tracking method: efficiency issues. *IEEE Transactions on Knowledge and Data Engineering* 7 (6):947-954, December.

Sagiv, Yehoshua, and Mihalis Yannakakis. 1980. Equivalences Among Relational Expresions with the Union and Difference Operators. *Journal of the Association for Computing Machinery* 27 (4):633-655, October.

Samuelson, Pamela. 1992. Copyright Law and Electronic Compilations of Data. *Communications of the ACM* 35 (2), February.

Schek, H. -J., and P. Pistor. 1982. Data Structures for an Integrated Data Base Management and Information Retrieval System. *8th International Conference on Very Large Data Bases*, 8-12 September 1982, in Mexico City, Mexico, pp 197-207.

Schek, H. -J., and M. H. Scholl. 1986. The Relational Model with Relation-Valued Attributes. *Information Systems* 11 (2):137-147.

Scholl, M. H. 1992. Extensions to the relational data model. In *Conceptual modelling, databases, and CASE: An integrated view of information systems development*, edited by L. P. and R. Zicari. New York: Jon Wiley & Sons.

SEC, U.S. Securities and Exchange Commission. 1999. Special Study: On-Line Brokerage: Keeping Apace of Cyberspace. U.S. Securities and Exchange Commission, http://www.sec.gov/news/studies/cyberspace.htm.

Shapiro, Carl, and Hal R. Varian. 1999. *Information Rules: A Strategic Guide to the Network Economy*. Boston, MA: Harvard Business School Press.

Shrager, Heidi J. 2001. E-Business: The Web @ Work/Zagat Survey. *Wall Street Journal*, 20 August 2001, Sec E-Business, p B6.

Sony v. Universal. 1984. *Sony Corp. v. Universal City Studios, Inc.*, U. S. Supreme Court 464:417.

Spaulding, Michelle L. 1998. *The doctrine of misappropriation* [Web]. Harvard Law School, 21 March 1998 [cited December 1999]. http://cyber.law.harvard.edu/metaschool/fisher/linking/doctrine/.

staff. 2000. *Federal judge says MP3.com willfully violated music copyrights*, 6 September 2000, 2:53P EDT [cited 4 January 2001].

http://www.cnn.com/2000/LAW/09/06/mp3.lawsuit.

Tabke, Brett. 1999. *PriceMan Sued by MySimon* [Web]. Saerch Engine World.com, 24 September 1999 [cited 11 December 2000 2000]. http://www.searchengineworld.com/news/lawsuit.htm.

Taylor, Chris, Peter Turner, Joe Cummings, and et al. 1997. *South-East Asia on a shoestring*. Ninth ed. Melbourne, Australia: Lonely Planet Publications.

Terry, Andrew. 1988. Misappropriation of a Competitor's Trade Values. *The Modern Law Review* 51:296, May.

Ticketmaster v. Microsoft. 1997. *Ticketmaster Corp. v. Microsoft Corp.*, 97:3055PP (settled).

Total News. 1997. *Washington Post Company v. Total News Inc.*, S. D. N. Y. 97:1190.

Transradio Press Service. 1937. *Twentieth Century Sporting Club, Inc. v. Transradio Press Service*, New York Supreme Court 300:159.

Tsur, D., J. Ullman, S. Abiteboul, C. Clifton, R. Motwani, S. Nestorov, and A. Rosenthal. 1998. Query flocks: A generalization of association-rule mining. *ACM SIGMOD International Conference on Management of Data*, June, in Seattle, WA, pp 1-12.

Tyson, L, and E Sherry. 1997. Statutory protection for databases: economic and public policy issues. Information Industry Association.

Tzafestas, Elpida. 2000. Toward Adaptive Cooperative Behavior. *Proceedings of the Simulation of Adaptive Behavior Conference*, September, in Paris, France.

U.S. v. Microsoft. 2001. *United States of America v. Microsoft Corporation*, U.S. Court of Appeals for the District of Columbia Circuit 253:34 (28 June).

Ullman, Jeffrey D. 1988. *Principles of database and knowledge-base systems, volume 1*. Principles of Computer Science, Edited by A. V. Aho and J. D. Ullman. 2 vols. Vol. 1, *Principles of Computer Science*. Rockville, Maryland: Computer Science Press.

Ullman, Jeffrey D. 1989. *Principles of database and knowledge-base systems, volume 2*. Principles of Computer Science, Edited by A. V. Aho and J. D. Ullman. 2 vols. Vol. 2, *Principles of Computer Science*. Rockville, Maryland: Computer Science Press.

Ullman, Jeffrey D., and Jennifer Widom. 1997. *A First Course in Database Systems*. New Jersey: Prentice-Hall, Inc.

Van de Sompel, Herbert, and Patrick Hochstenbach. 1999. Reference linking in a hyrid library environment. *D-Lib Magazine* 5 (4), http://www.dlib.org/dlib/april99/van_de_sompel/04vande_sompel-pt1.html.

Van Gelder, Allen, and Rodney Topor. 1991. Safety and Translation of Relational Calculus Queries. *ACM Transactions on Database Systems* 16 (2):235-78, June.

Walsh, N. 1997. Introduction to XML. *XML Journal* 2 (4), Fall.

Wang, Richard, and Stuart Madnick. 1990. A Polygen Model for Heterogeneous Database Systems: The Source Tagging Perspective. *16th International Conference on Very Large Data Bases (VLDB)*, 13-16 August, in Brisbane, Australia.

Warren Publishing v. Microdos. 1997. *Warren Publishing, Inc. v. Microsods Data Corp*, United States Court of Appeals, Eleventh Circuit 93:8474 (10 June).

Wiederhold, G. 1992. Mediators in the architecture of future information systems. *IEEE Computer* 25 (3):38-49, March.

Winokur, Marilyn. 1999. H.R. 354 Testimony on behalf of Thomson Corporation and the Coalition Against Database Piracy. Before *Subcommittee on Courts and Intellectual Property of the Committee on the Judiciary*, US House of Representatives, Washington, DC. 18 March 1999. http://www.house.gov/judiciary/106-wino.htm.

Wolverton, Troy. 2000. *Judge bars Bidder's Edge Web crawler on eBay*. CNET News.com, 25 May 2000, 12:30 PST [cited 3 December 2000]. http://news.cnet.com/news/0-1007-200-1948171.html.

Wong, Stephanie. 1999. Estimated $4.35 billion in ecommerce sales at risk each year. Zona Research, Inc., http://www.zonaresearch.com/info/press/99-jun30.htm.

Woodruff, Allison, and Michael Stonebraker. 1997. Supporting fine-grained data lineage in a database visualization environment. *Proceedings of the 13th International Conference on Data Engineering*, April, in Birmingham, England, pp 91-102.

Zuckerman, Gregory, and Rebecca Buckman. 1999. Data Providers Face Internet Challengers. *Wall Street Journal*, 21 September, p C1.