Schema Merging and Mapping Creation for Relational Sources

Rachel Pottinger
University of British Columbia
201-2366 Main Mall
Vancouver, BC V6T 1Z4, Canada
rap@cs.ubc.ca

Philip A. Bernstein
Microsoft Research
One Microsoft Way
Redmond, WA 98052-6399, USA
philbe@microsoft.com

ABSTRACT

We address the problem of generating a mediated schema from a set of relational data source schemas and conjunctive queries that specify where those schemas overlap. Unlike past approaches that generate only the mediated schema, our algorithm also generates view definitions, i.e., source-to-mediated schema mappings.

Our main goal is to understand the requirements that a mediated schema and views should satisfy, such as completeness, preservation of overlapping information, normalization, and minimality. We show how these requirements influence the detailed structure of schemas and view definitions that are produced. We introduce a normal form for mediated schemas and view definitions, show how to generate them, and prove that schemas and views in this normal form satisfy our requirements.

The view definitions in our normal form use stylized GLAV mappings, for which query rewriting is easier than general GLAV mappings. We demonstrate the efficiency of query rewriting in a prototype implementation.

1. INTRODUCTION

In data integration, users query multiple sources using a unified, mediated schema rather than querying each source separately. Each query over the mediated schema is then translated into queries over the source schemas. The results of these queries are combined and returned to the user.

To enable this scenario, the system needs view definitions, that is, mappings that relate the mediated schema to its data sources. In this paper, we analyze what constitutes a good mediated schema and mappings and show how to obtain them.

We focus on the case where one starts with a set of data source schemas. Although the disjoint union of the data source schemas could be used as the mediated schema, this is usually not what is wanted because it is highly redundant. The reason why a mediated schema over multiple data sources makes sense is that the data sources are closely related. Thus, some information is represented in two or more schemas, often in different ways. Even if a user were willing to deal with this redundancy, the complexity of accessing the information in different representations would make the schema hard to use. Therefore, such overlapping information should have just one representation in the mediated schema with a mapping to each of its representations in the source schemas.

One step in the development of the mediated schema is identifying the overlapping source-schema elements that should be collapsed. This is the database designer's job—to identify those overlapping elements, possibly with the help of a schema matching tool. The designer needs to explain how each overlapping element is mapped to each of the data sources in which it appears.

> **Example 1** Suppose we want a mediated schema to integrate two travel databases, Go-travel and Ok-travel. Go-travel has three relations:
>
> Go-flight(f-num, time, meal)
> Go-price(f-num, date, price)
> Go-airline(airline, phone)
>
> The attribute f-num is the flight number and meal is a boolean. The other attributes are self-explanatory. Ok-travel has just one relation: Ok-flight(f-num, date, time, price, nonstop), where nonstop is a boolean. The overlapping information in Ok-travel's and Go-travel's schemas could be represented in a mediated schema by Flight(f-num, date, time, price). □

The overlapping elements are only part of the mediated schema. Some elements that are unique to a particular data source are also passed through to the mediated schema. For example, if the schemas are all relational, then a data source may have a relation R that does not overlap any relation in any other data source and should be made part of the mediated schema. In addition, even if R does overlap a relation R′ of another data source, R may have an attribute that does not appear in R′ or any other data source and that may be valuable to pass through to the mediated schema.

In the previous paragraph, to be technically precise we should be calling R a relation schema, not a relation. However, since this paper is focused on schemas more than data, we usually use "relation" to mean a relation schema. We use the more precise terminology only when the meaning is not clear from the context.

Suppose one wants the mediated schema to expose all of the information in the data sources. In that case, the mediated schema should include all of the overlapping schema elements, plus all source-specific elements. By source-specific, we mean that the schema elements are not subsumed by overlapping elements.

EDBT'08, March 25-30, 2008, Nantes, France.

Example 2 Continuing with Example 1, suppose we want to expose all of Ok-travel's and Go-travel's information in the mediated schema. In addition to the overlapping information in Flight(f-num, date, time, price), we see that OK-travel has source-specific information about flights being nonstop, which does not appear in Go-travel's schema. There are two choices on how to include nonstop. We can pass through Ok-flight as a separate relation in the mediated schema. However, since we already have most of Ok-flight's attributes in the Flight relation, it seems more natural to add nonstop to Flight, yielding Flight(f-num, date, time, price, nonstop). A similar issue arises with meal in Go-flight.

Go-travel also has source-specific data about airline phone numbers, so we need to pass through Go-airline(airline, phone). We could handle this just like nonstop, by adding the attributes airline and phone to Flight. However, this would represent two independent types of information in the same relation, information about flights and about airlines. This violates standard database design principles, which say that independent relationships should be represented in different relations. It is better simply to add a relation Airline(airline, phone) to the mediated schema. □

So far, we have identified two issues that affect the design of mediated schemas: (1) overlapping information should have a unique representation in the mediated schema; and (2) source-specific schema information can be passed through to the mediated schema either by extending relations that represent the overlapping information or by adding relations, depending on whether the source-specific information is or is not closely dependent on the overlapping information. Now let us look at the mapping between the mediated schema and source schemas.

The database designer needs a language in which to express overlapping schema elements. If the data sources are relational, a natural choice is relational queries. The overlap is defined by a set of queries, one over each data source, which return type-compatible data.

In what follows, we frequently refer to global-as-view (GAV), local-as-view (LAV), and global-local-as-view (GLAV) mappings. We assume a basic understanding of these concepts. A recent survey appears in [10].

Example 3 The overlapping flight information in Example 1 could be expressed by the following Datalog queries:

```
Flight(f-num, date, time, price) :- Go-flight(f-num, time, meal),
                                    Go-price(f-num, date, price)
Flight(f-num, date, time, price) :- Ok-flight(f-num, date, time,
                                    price, nonstop)
```

The Flight relation is a view of Go-travel and Ok-travel. The above queries comprise a simple GAV mapping. The left side of the mapping is a relation in the mediated schema, and the right side is a query over the data sources. Hence these queries appear to be the mapping we need between the mediated schema Flight and the data sources.

However, if we pass through Go-flight.meal and Ok-flight.nonstop to the mediated schema, we obtain Flight(f-num, date, time, price, meal, nonstop). This situation is more complex. One problem is that Ok-flight does not have a meal attribute. Therefore, we are driven to represent the mapping between Flight and Ok-flight using a LAV mapping: Ok-flight(f-num, date, time, price, nonstop) :- Flight(f-num, date, time, price, meal, nonstop). (More details are in Section 4.2.) This mapping is different than what the database designer specified in the second query above. Combined with the first mapping, the overall mapping between the mediated schema and data sources uses GLAV. □

Past work on mediated schema creation has focused on identifying and collapsing overlapping elements. This is the problem of *schema merging*. Certainly, schema merging is an important ingredient. However, as the examples above show, there is more to it. We need to pass through some source schema elements to the mediated schema. There may be more than one way to do this, and the resulting mapping may be more complex than a simple GAV query. In this paper, we generalize these observations into requirements for mediated schema and mapping design and an algorithm for obtaining them. We call this problem **semantic merge**.

The semantic merge problem is the following: Given mapping expressions that define the overlapping parts of a set of relational schemas that represent data sources, (i) generate a mediated schema that collapses these overlapping elements according to a given specification and passes through source schema elements as appropriate and (ii) generate mappings between the mediated schema and data sources. Our contributions are as follows:

- We propose conjunctive queries as a way of expressing overlapping information in data sources (Sections 2.2 and 2.3)
- We define technical requirements for mediated schema and mapping design (Section 2.4).
- We define a normal form for mediated schemas and mappings (Sections 3.1 and 3.2), prove that it satisfies the technical requirements (Section 3.4), and comment on its properties (Section 4).
- We give an algorithm that generates normal-form mediated schemas and mappings (Section 3.3).
- We report on an implementation of the algorithm (Section 5).

We discuss related work in Section 6. Section 7 is the conclusion.

2. REQUIREMENTS

2.1 Introduction

The motivation and examples in Section 1 lead to the following five criteria that we propose as requirements for a mediated schema and mapping to satisfy:

i. Completeness: All information in the source schema should be exposed in the mediated schema.

ii. Overlap preservation: Each of the overlapping elements specified in the input mapping is exposed in a mediated schema relation.

iii. Extended overlap preservation: Source-specific elements that are associated with a source's overlapping elements are passed through to the mediated schema.

iv. Normalization: Independent entities and relationships in the source schemas should not be grouped together in the same

relation in the mediated schema. In particular, source-specific schema elements should not be grouped with overlapping schema elements if the grouping co-locates independent entities or relationships.

v. Minimality: If any elements of the mediated schema are dropped then the mediated no longer satisfies (i) – (iv) above

We do not claim that users will want their mediated schema and mapping to satisfy all of these criteria in all scenarios. However, we do claim that these criteria are often desirable and that it is worthwhile to understand how these criteria influence the choice of mediated schema and mapping. We will have more to say about this in Section 4**.** We also show how these criteria correspond to hand-crafted mediated schemas in Section 5.

To make the above requirements technically precise, we start by defining the language in which to express overlapping parts of the source schemas. Simple correspondences between elements are not enough, because we need a formal semantics of the overlapping parts to guide the development of an output mapping. We choose conjunctive queries for this purpose, which are defined in Section 2.2. They are expressive enough to demonstrate our design principles. Additionally, they comprise the mapping language that is most commonly used in the research literature on data integration (e.g., [8]). We explain how to use conjunctive queries to express overlapping schema elements in Section 2.3.

Using this mapping language, we then define technical requirements for a mediated schema and mapping that correspond to the intuition developed in Section 1.

2.2 Conjunctive Queries

We express mappings over relational schemas as conjunctive queries using Datalog notation, as in Example 3. A **database schema** is a set of relation schemas. Each **relation schema** R has a relation name and a sequence of attribute names, denoted attr(R). The **arity** of a relation schema is the number of attribute names it has.

A **conjunctive query** Q has the form $q(X)$:- $e_1(X_1)$, …, $e_n(X_n)$, where q and e_1, ..., e_n are relation names. The **subgoals** $e_1(X_1)$, …, $e_n(X_n)$ are collectively the **body** of Q, denoted body(Q). The predicate that appears on the left side of the query, q(X), is called the **head** of query. The predicate name q of the head is the **Intensional Database (IDB)** of Q, denoted IDB(Q). The tuples X, X_1, …, X_n have the same arity as the relations in which they appear. They contain variables or constants. The query Q must be **safe**, meaning that every variable in X also appears in body(Q). Vars(Q) refers to the variables of Q. The variables in the body of Q but not the head (i.e., Vars(Q)-X) are called **existential** variables. The **answer** to query Q is an assignment of constants to the variables X such that for some assignment of constants to Q's existential variables, body(Q) is true. The answer to a set of queries with a common head is the union of the answers of the queries in the set. A conjunctive query has the same expressive power as a SQL select-project-join query without arithmetic comparisons.

Example 4 The following query asks for the prices of flights that are listed in both Go-travel and Ok-travel:

Q(f,p) :- Go-price(f,d,p), Ok-flight(f,d,t,p,n)

Vars(Q) = {f, p, d, t, n}. Variables d, t and n are existential. □

Let Q_1 and Q_2 be two sets of queries whose heads have the same arity. We say that Q_1 is **contained in** Q_2, denoted $Q_1 \subseteq Q_2$, if the answer to Q_1 is a subset of the answer to Q_2 for *all* database instances. Q_1 and Q_2 are **equivalent** if $Q_1 \subseteq Q_2$ and $Q_2 \subseteq Q_1$, i.e., they produce the same answer for every given database.

2.3 Conjunctive Mappings

A **mapping** is a relationship between the instances (i.e., the states) of two schemas. Formally, let Inst(S) be the set of all instances of a schema S. Then a mapping between database schemas U and V is a subset of Inst(U) × Inst(V). In this paper, we use conjunctive queries to define mappings.

Let **S** = {S_1, ... S_n} be a set of database schemas, one for each data source. We express each overlapping part of a set of source schemas by a set of two or more conjunctive queries with a common IDB, where the body of each query is defined over one schema in **S**. This is called an **overlap specification** (or, simply, an **overlap**). For example, the queries in Example 3 comprise a overlap specification over **S** = {Go-travel, Ok-travel}, where Flight is the common IDB. We interpret each query in an overlap using open world semantics, i.e., it computes a subset of the tuples satisfied by the head.

Subgoals e_i, e_k of query Q are **connected** if e_i has a variable in common with e_k or e_i is connected to another subgoal of Q that is connected to e_k. For every query Q in an overlap, all subgoals of Q must be connected. That is, Q has no Cartesian products.

To simplify the case analysis of definitions and theorems that follow, we do not allow constants to appear in conjunctive mappings. None of our technical results depend on this assumption.

To avoid having to rename attributes and relations in the mediated schema, we require that overlap specifications adhere to naming conventions. The conventions do not affect the expressive power of mappings; any set of conjunctive queries can be made to satisfy them just by renaming variables. The conventions are as follows

i. In an IDB $q(a_1, ..., a_m)$, each a_i ($1 \le i \le m$) is called a **variable position**. For each IDB name q in a set of overlap specifications and for each variable position of q, the same variable name must be used in all appearances in that variable position. For example, the following overlap violates this requirement because name and aname appear in the only variable position of Airline.

   Airline(name) :- TravelOn-airline (name, phone)
   Airline(aname) :- MyTravel-airline(aname, address)

   This convention allows us to use, without ambiguity, the variable name in each variable position as the name of the corresponding attribute in the mediated schema.

ii. For a given IDB name in an overlap, an existential variable name may appear in at most one conjunctive query with that IDB name. For example, the following overlap violates this requirement because phone is existential in both queries:

   Airline(name) :- TO-airline(name, phone)
   Airline(name) :- H-airline(name, phone, fax).

   Like the previous convention, this allows us to use the name of each existential variable as the name of the corresponding attribute in the mediated schema.

iii. The relation names in **S** and the IDBs of overlaps are distinct. That is, for each database schema S_i in **S** if relation name R appears in S_i, then R does not appear in any other schema S_k ($i \neq k$) or as an IDB in any overlap.

An overlap specification that conforms to the above conventions is said to be **well-formed**.

2.4 Technical Requirements

In this section, we make the correctness criteria of Section 2.1 more precise: completeness, overlap preservation, extended overlap preservation, normalization, and minimality.

2.4.1 Completeness

We want to ensure there is no information loss in the mediated schema. We can do this with the following **completeness** requirement: for each source relation R there is a query over the mediated schema that is equivalent to the identity query over R. This corresponds to the notion of query dominance in Hull's information capacity model [9], which is a common way to judge the information preservation of one schema with respect to another [17].

This completeness criterion ensures that each source relation is accessible by a query over the mediated schema. But it says nothing about how complex that query might need to be. To ensure the mediated schema is understandable and easy to use, we strengthen completeness by requiring that the query over the mediated schema M refers to only one relation in M. That is, we require that for each source relation R there is a query over one relation in the mediated schema that is equivalent to the identity query over R.

This requirement implies that for each relation R in a source, there is a corresponding relation in the mediated schema that has all of the attributes of R and possibly others. It also says something about the mapping between R and the mediated schema, namely that all data from the sources can be accessed and that similarly structured data from different sources can be distinguished. For example, suppose that in addition to the relation Go-airline(airline, phone) in Go-travel there is a relation Ok-airline(airline, phone) in Ok-travel. Then it is not enough to include a relation Airline in the mediated schema defined as follows:

```
Airline(airline, phone) :- Go-airline(airline, phone)
Airline(airline, phone) :- Ok-airline(airline, phone)
```

because only the union of the two relations Go-airline and Ok-airline can be queried in the mediated schema. We will show how to create mediated schemas that avoid this problem in Section 3.

2.4.2 Overlap preservation

Overlap preservation requires that each of the overlapping elements specified in the input mapping is exposed in a mediated schema relation. Overlapping elements are defined by an overlap, which is a set of conjunctive queries. Therefore, this requirement can be stated in an analogous fashion to completeness, as follows: For each overlap, there is a query Q over one relation in the mediated schema that is equivalent to the overlap.

2.4.3 Extended overlap preservation

To satisfy completeness, we may want to add attributes to a mediated schema relation beyond those that are needed for overlap preservation. For example, consider the second query in the overlap specification of Example 3:

```
Flight(f-num, date, time, price) :- Ok-flight(f-num, date,
                                    time, price, nonstop)
```

Overlap preservation implies that there is a relation in the mediated schema that includes the attributes f-num, date, time, price. Completeness requires that there is a query over one relation in the mediated schema that is equivalent to the identity query over Ok-flight. Since nonstop is the only attribute of Ok-flight that does not appear in Flight, one way to satisfy completeness is simply to add nonstop to the mediated schema relation that includes Flight (we will call it M-Flight). This seems more economical than, and hence preferable to, adding Ok-flight to the mediated schema in addition to Flight.

Another reason to add attributes to a mediated schema is convenience. For example, consider the first query in the overlap specification of Example 3**:**

```
Flight(f-num, date, time, price) :- Go-flight(f-num, time, meal),
                                    Go-price(f-num, date, price)
```

The attribute meal appears in Go-flight but not in Flight. By including meal in M-Flight, we enable the user to query this information without performing a join. Given that meal already appears in a source relation (namely, Go-flight) with the other attributes of Flight, we know that it has a strong relationship with those attributes. Therefore, including it in M-Flight seems like a worthwhile convenience.

Unlike the previous example, we cannot claim completeness as a reason to include meal in M-Flight. One might think that meal would help us with completeness (along with the right mapping), because a projection query on M-Flight over f-num, time, meal would return the content of Go-flight. But this is incorrect. It only returns the subset of Go-flight that joins with Go-price, which does not help us with respect to completeness.

Therefore, whether or not we include meal in M-Flight, for completeness we will need another mediated schema relation R such that a query over R is equivalent to Go-flight(f-num, time, meal). Thus, from a completeness standpoint, including meal in M-Flight is unnecessary. Still, from a convenience standpoint, it is desirable to include it, to avoid requiring a join with R to associate meal with the other attributes of M-Flight. The convenience attribute might even be a join variable, as in the following overlap specification:

```
Flight(date, price, nonstop) :- My-flight(time, f-num, nonstop),
                                My-price(date, time, f-num, price)
Flight(date, price, non-stop):- Auction-Flight(date, price, nonstop)
```

For a flight in My-flight, it would be handy to get the flight number if you can get it (and the time, which is also existential) without performing a join.

We capture the convenience aspect of adding attributes to an overlap as follows. If Q is a query in an overlap and has an existential variable, then we define the **extended overlap query** of Q to be a query Q′ whose IDB is augmented with all of the existential variables in body(Q). We then add the requirement that for each extended overlap query Q′, there is a query over one relation in the mediated schema that is equivalent to Q′.

2.4.4 Normalization

Taking the logic of extended overlaps to the extreme, one could argue to make the mediated schema a universal relation, i.e., one relation that includes all of the attributes of all of the source relations. This is not the problem we are addressing. Sidestepping a debate about the merits of a universal relation, we simply remind the reader that our goal is to generate a mediated schema that collapses these overlapping elements according to a given specification. We therefore need a principle that limits the amount of source schema that is collapsed.

One such principle could be that the mediated schema should satisfy fourth normal form (or pick your favorite stronger normal form). However, we do not recommend this because a user may want a mediated schema that violates a normal form. For example, in the overlap above, it might be that f-num is the key of Go-flight and (f-num, date) is the compound key of Go-price. So Flight violates second normal form. The main justification for normal forms relates to update behavior. We see no reason to prohibit Flight from appearing in a mediated schema that is used only for queries.

We therefore define a weaker criterion whose goal is to avoid causing normalization violations beyond those introduced by the overlap specifications. The criterion is that for each mediated schema relation R that corresponds to an overlap specification O, every attribute of R appears in the head or a body of O. By R **corresponds to** O, we mean that R includes the attributes of IDB(O) and the query that projects R on those attributes is equivalent to O.

2.4.5 Minimality

We require that the mediated schema cannot be made smaller and still satisfy completeness, overlap preservation, extended overlap preservation, and normalization. We say that database schema V is **minimal** with respect to property P if there is no database schema U satisfying P such that:

1. For all relations $R \in U$ there exists a relation $R' \in V$ such that $attr(R) \subseteq attr(R')$, and
2. For some relation $R \in V$, $R \notin U$ or $attr(R) \subset attr(R')$.

Intuitively, the above definition says that V is minimal with respect to P if there is no smaller schema U satisfying P (condition 1) that can be derived from V by deleting a relation from V or deleting an attribute of a relation of V (condition 2).

2.4.6 Summary

In Section 2.1 we presented informal requirements for a mediated schema and mapping. In the rest of Section 2, we formalized these requirements, which we restate here. The input to mediated schema creation consists of relation schemas for the data sources and a set of overlap specifications, which are conjunctive queries. The output is a relational mediated schema and a mapping between the mediated schema and data sources that satisfies the following **mediated schema criteria**:

i. Completeness: For each source relation R, there is a query over the mediated schema that is equivalent to the identity query over R.

ii. Overlap preservation: For each overlap, there is a query over one relation in the mediated schema that is equivalent to the overlap.

iii. Extended overlap preservation: For each extended overlap query Q, there is a query over one relation in the mediated schema that is equivalent to Q.

iv. Normalization: For each mediated schema relation R that corresponds to an overlap specification O, every attribute of R appears in the head or a body of O.

v. Minimality: The mediated schema cannot be made smaller and still satisfy (i) – (iv) above.

3. MEDIATED SCHEMA NORMAL FORM

For a given set of source schemas **S** and a set of overlap specifications **O**, we define a normal form for a mediated schema M over **S** and **O** (in Section 3.1) and for a conjunctive mapping map_{MS} between M and **S** (in Section 3.2). In Section 3.3, we show that this normal form satisfies the mediated schema criteria.

3.1 The Mediated Schema

In what follows, we often use the same relation names in the mediated schema and data sources, when they obviously correspond. To distinguish between them, we use the prefix "M." for relation names in the mediated schema, M.

The mediated schema criteria give us two main reasons to create a relation M.R in the mediated schema:

a. Overlap Relation – M.R is derived by applying overlap preservation and extended overlap preservation to each overlap specification, mitigated by normalization. More concretely, R is the IDB of an overlap specification O and M.R's attributes are the set of all variables in the bodies of all queries in O. We call M.R an **overlap relation** and say that M.R **corresponds to** O.

b. Completeness Relation - R is a source relation whose content is not equivalent to a query over the relations defined by (a). In this case, add M.R to the mediated schema. We call M.R a **completeness relation** and say that M.R **corresponds to** R.

If a mediated schema conforms to rules (a) and (b), we say it is in **mediated schema normal form** (**MSNF**).

Example 5 Reconsider the overlap specification in Example 3

```
Flight(f-num, date, time, price) :- Go-flight(f-num, time, meal),
                                    Go-price(f-num, date, price)
Flight(f-num, date, time, price) :-
                                    Ok-flight(f-num, date, time, price, nonstop)
```

The mediated schema should include an overlap relation M.Flight(f-num, date, time, price, meal, nonstop), because Flight is the IDB of the overlap and {f-num, date, time, price, meal, nonstop} is the set of all variables in the bodies of all queries in the overlap. With a suitable mapping (which we have not defined yet), it is possible that a query on M.Flight that projects f-num, date, time, price, and nonstop would be equivalent to the content of Ok-flight. Therefore, case (b) does not apply to Ok-flight.

The same approach would not work for Go-flight(f-num, time, meal), since the body of the first query in the overlap includes a join with Go-price. So a query on M.Flight that projects f-num, time, and meal would return the subset of Go-flight that

joins with Go-price. Therefore, following (b) above, we need to include completeness relations M.Go-flight(f-num, time, meal) and M.Go-price(f-num, date, price) in the mediated schema. Due to (b), we also need to include M.Go-airline(airline, phone), which is in the schema in Example 1 but not in any overlap at all. □

Example 5 suggests that we need to add a source relation to M unless it appears alone in the body of a query in an overlap. But this is not quite enough, as the following example shows.

Example 6 Suppose we add the following query to the overlap specification of Example 3:

Flight(f-num, date, time, price) :- US-flight(f-num,date,time,price)

The mediated schema would still be M.Flight(f-num, date, time, price, meal, nonstop). A query on M.Flight that projects f-num, date, time, and price would return a relation that includes the content of US-Flight. But it would also include the projection of Ok-flight on f-num, date, time, and price and of Go-flight(f-num, time, meal) joined with Go-price(f-num, date, price). Thus, (b) tells us to add M.US-flight(f-num, date, time, price) to M. □

We say that a source relation R **is subsumed by** an overlap O if R appears alone in the body of a query Q in O and R has at least one existential variable in Q. In Example 5 Ok-flight is subsumed by Flight, since it appears alone in the second query of the overlap and has an existential variable nonstop. As was shown for Ok-flight in Example 5, when R is subsumed by an overlap O, case (b) above does not apply. Notice that it is important that O is well-formed (see Section 2.3), so that the existential variable is uniquely named. This ensures that the projection query on attr(R) applied to the completeness relation returns the tuples of R and no other source relation.

This property of the projection query also explains why we defined an extended overlap query only for cases where there is an existential variable in the body of the overlap query. If there is no existential variable, then the projection of the overlap relation on the attributes returns tuples in addition to those of R and hence doesn't help us with respect to completeness.

Given the definition of subsumed, we can now restate case (b):

b. Completeness Relation – If R is a source relation that is not subsumed by an overlap, then add R to the mediated schema.

Suppose two completeness relations have the same set of attributes, such as the example

Airline(airline, phone) :- Go-airline(airline, phone)
Airline(airline, phone) :- Ok-airline(airline, phone)

that we saw at the end of Section 2.4.1.These could be combined into a single mediated schema relation by adding a tag attribute, such as Airline(airline, phone, tag). For each tuple, the tag identifies which source relation(s) contain the relevant tuple. This is a valid alternative to our completeness relations. However, it makes the mediated schema less self-describing by hiding the corresponding source in the value of the tag attribute. For clarity, we therefore use completeness relations in what follows.

Notice that according to our definition of minimality, the database schema {Airline(phone, airline, tag)} does not contradict the minimality of the database schema {Go-airline(phone, airline), Ok-airline(phone,airline), Airline(phone, airline)}, because the former cannot be obtained from the latter by deleting relations and/or attributes. Thus, our notion of minimality is that of a local minimum, not a global minimum.

3.2 The Mapping between Mediated and Source Schemas

In Example 3 we showed that the use of an extended overlap in the mediated schema made it desirable to use a GLAV mapping between a mediated schema and data sources. In this section, we define a stylized GLAV mapping between M and **S**, called an MSNF mapping, which has two nice properties: it ensures the mediated schema and mapping satisfy the mediated schema criteria; and it ensures that it is easy to produce exact rewritings of the queries required by completeness, overlap preservation, and extended overlap preservation.

Let M be an MSNF schema derived from source schemas **S** and overlap specifications **O**. The MSNF mapping map_{MS} between M and **S** is defined using a schema I, called the **intermediate schema**, that is distinct from M and **S**. Schema I is a "helper schema" that sits between M and **S**. The relations of I with respect to (w.r.t.) M and **O** are (1) copies of the completeness relations in M and (2) the heads of the extended overlap queries of all overlaps in **O**. The latter relations are named by adding to the IDB of the overlap a subscript that is the index of the query in the overlap. For example, I would include the following two relations for the overlap in Example 3: $I.Flight_1$(f-num, date, time, price, meal) and $I.Flight_2$(f-num, date, time, price, nonstop). More precisely:

i. For each completeness relation M.R in M, there is a relation I.R′ in I that has the same attributes as M.R. We say that $I.R_i$ **corresponds to** M.R.

ii. For each overlap O in **O**, for each query Q_i in O ($1 \le i \le |O|$) there is a relation $I.R_i$ in I where R = IDB(Q_i) and $I.R_i$ has the same attributes as body(Q_i). We say that $I.R_i$ **corresponds to** Q_i in O .

The **MSNF mapping** map_{MS} between M and **S** has LAV and GAV queries for completeness relations and for extended overlap queries, both of which are expressed using the intermediate schema I. They are defined as follows:

a. LAV:

1. For each completeness relation M.R in M, map_{MS} includes the following query:
   I.R(attr(M.R)) :- M.R(attr(M.R)).

2. For each relation $I.R_i$ in I that corresponds to some overlap O in **O**, map_{MS} includes the following query: $I.R_i$(attr($I.R_i$)) :- M.R(attr(M.R)) where M.R corresponds to O.

b. GAV:

1. For each completeness relation I.R in I, map_{MS} includes the following query:
   I.R(attr(I.R)) :- R(attr(I.R)).

```
Procedure MSNFSchemaMerge(S, O)
// S is the set of schemas to be merged
// O is a set of overlaps between them
M = ∅   //the merged schema to create
Let R = {r ∈ S | r is not subsumed by an overlap in O}
For each relation r ∈ R
   Let m be a new relation
   name(m) = M.name(r)
   attr(m) = attr(r)
   Add m to M
For each IDB name q ∈ IDB names in O
   Let m be a new relation
   Let Vars_q be the duplicate-free union of the variables
                 of queries that define q in O
   name(m) = name(q)
   attr(m) = Vars_q
   Add m to M
Return M
```

```
Procedure MSNFMappingCreation(S, O, M)
// S is a set of schemas, O is an overlap
// M is the output from MSNFSchemaMerge(S, O)
LAV-views_M = ∅
GAV-views_M = ∅
For each relation m ∈ M
   If e ∈ S and e corresponds to m
      Let q be a fresh IDB name
      //  i.e., q is an IDB name that does not appear as an
      //  IDB name elsewhere in O or map_MS.
      Let lav_m  = q(attr(m)) :- m(attr(m))
      Let gav_m = q(attr(m)) :- e(attr(e))
       // The relations m and e in the definition of lav_m
       //  and gav_m respectively are the same (from the
       //  definition of MSNFSchemaMerge).
       //  Similarly, attr(m) = attr(e).
      Add lav_m to LAV-views_M
      Add gav_m to GAV-views_M
    For each overlap query oq ∈O
      Let cname = IDB(oq)
      Let m be the relation in M such that cname
           corresponds to m
      Let q be a fresh IDB name
      lav_c  = q(Vars(oq)) :- m(attr(m))
      gav_c = q(Vars(oq)) :- body(oq)
      Add lav_c to LAV-views_M
      Add gav_c to GAV-views_M
Return LAV-views_M and GAVviews_M.
```

Figure 3-1 Algorithms to generate an MSNF schema and mapping

2. For each relation $I.R_i$ in I that corresponds to some query Q_i in overlap O in **O**, map_{MS} includes the following query: $I.R_i(attr(I.R_i))$:- body(Q).

Cases (a1) and (b1) apply to completeness relations. Each completeness relation is identical to its corresponding source relation and intermediate relation. For example, the source relation Go-flight(f-num, time, meal) has a corresponding mediated schema relation M.Go-flight(f-num, time, meal) (see Example 5) and a corresponding intermediate schema relation I.Go-flight(f-num, time, meal). Therefore, by (a1), map_{MS} includes the LAV query I.Go-flight(f-num, time, meal) :- M.Go-flight(f-num, time, meal). And by (b1), map_{MS} includes the GAV query I.Go-flight(f-num, time, meal) :- Go-flight(f-num, time, meal).

Cases (a2) and (b2) apply to extended overlap queries. Consider the i^{th} query Q_i in an overlap O whose IDB is R. The head of Q_i's overlap or extended overlap query has IDB R and its attributes are A_i = $attr(body(Q_i))$. The corresponding intermediate schema relation is $I.R_i(A_i)$. Hence, Q_i's GAV query in (b2) is its overlap or extended overlap query with a slightly different IDB, namely $I.R_i$. By contrast, the mediated schema relation M.R that corresponds to O has the union of attributes in the heads of all overlap queries and extended overlap queries for O. So in general, M.R's attributes are a proper superset of A_i. Hence the LAV query for Q_i's overlap or extended overlap query is a projection of M.R on A_i. For example, consider the second query (call it Q_2) in the overlap of Example 5: Flight(f-num, date, time, price) :- Ok-flight(f-num, date, time, price, nonstop). The corresponding intermediate schema relation is $I.Flight_2$(f-num, date, time, price, nonstop). The corresponding extended overlap relation in the mediated schema is M.Flight(f-num, date, time, price, meal, nonstop); it includes meal which comes from the other overlap query in Example 5. So the LAV query for $I.Flight_2$ is $I.Flight_2$(f-num, date, time, price, nonstop) :- M.Flight(f-num, date, time, price, meal, nonstop), and the GAV query for Q_2 is $I.Flight_2$(f-num, date, time, price) :- Ok-flight(f-num, date, time, price, nonstop).

3.3 An Algorithm to Generate MSNF

The definitions of MSNF schema and mapping are, in effect, algorithms that generate the schema and mapping. For completeness, we restate them in a more procedural format in Figure 3-1.

3.4 Correctness

In this section, we prove that every MSNF schema and mapping satisfy the mediated schema criteria of Section 2.4. A reader can skip this section without loss of continuity.

Theorem 1: Let **S** be a set of source schemas and **O** a set of overlap specifications. Every MSNF mediated schema M over **S** and **O** and mapping map_{MS} satisfy the mediated schema criteria. □

We prove Theorem 1 in Lemmas 2-6 below, showing that each of the five mediated schema criteria is satisfied. To do this, we need to weaken slightly the notion of query equivalence that appears in the mediated schema criteria. Since MSNF uses GLAV mappings to relate M and **S**, we instead use maximally-contained rewritings, which return the maximal set of sound answers that can be obtained given the sources in **S**, which are not assumed to be complete [8]. For example, completeness says that "For each source relation R, there is a query over the mediated schema that is equivalent to the identity query over R." Instead, we will show there is a query over the mediated schema whose maximally-contained rewriting is the identity query over R.

We start with a technical result about maximally-contained rewritings of projection queries over the mediated schema. This is used in the proof since the queries required for completeness and overlap preservation are projection queries. The result says that to

rewrite a projection query Q over M.R to be a query over I, replace it by the union of the set of all queries whose head is Q and whose body is an intermediate relation that includes all of Q's attributes and that has an associated LAV view whose body is R.

Lemma 1: Let M be an MSNF schema derived from source schemas **S** and overlap specifications **O**. Let I be the intermediate schema w.r.t. M and **O**. Let map_{MI} be the LAV part of the MSNF mapping between M and **S**. Let Q be a projection query over a relation M.R such that the variables of M.R in Q match the names of the attributes of M.R. Let $\{I_1, \ldots, I_n\}$ be the set of all intermediate relations I_i such that (i) the body of I_i's LAV view is M.R(attr(R))[1], and (ii) Vars(head(Q)) ⊆ Vars(head(I_i)). Then $\cup_{1\le j\le n}$ (head(Q) :- I_j(attr(I_j)) is a maximal rewriting of Q w.r.t. map_{MI}.

Proof: Let Q be "Q(A′) :- M.R(A′)". By [12], answering queries using views can be used to construct a maximally-contained rewriting of Q; we consider the MiniCon algorithm which produces maximally-contained rewritings (see proof in [19]). Due to the simple structure of the LAV views and the fact that Q has only one subgoal, the MiniCon algorithm is nearly trivial. There is only one subgoal to consider, which is for M.R. Each MiniCon Description (MCD) thus contains I_i for each LAV view whose body is M.R(attr(R)); since the query is a projection query, the head homomorphism is the identity homomorphism. Since the query is one subgoal long, the combination step will simply return the MCD. The resulting query is $\cup_{1\le j\le n}$ (Q :- I_j), as desired. □

Lemma 2: (Completeness): For each source relation R, there is a query over the mediated schema whose maximally-contained rewriting is the identity query over R.

Proof: There are two cases, depending on whether R is subsumed by an overlap. Case (1): If R is not subsumed by an overlap, then it appears in M. We show that the identity query over R is a maximally-contained rewriting of the identity query over M.R.

Let Q(attr(M.R)) :- M.R(attr(R)) be the identity query over M.R. There is only one view in map_{MS} that refers to M.R, a LAV view I(attr(M.R)) :- M.R(attr(M.R)). By Lemma 1, Q(attr(M.R)) :- I(attr(M.R)) is a maximally-contained rewriting of Q. The only other view that mentions I is the GAV view I(attr(R)) :- R(attr(R)). By unfolding the GAV view into Q, we conclude that Q(attr(M.R)) :- I(attr(M.R)) is equivalent to Q(attr(M.R)) :- R(attr(R)) as desired.

Case (2): If R is subsumed by an overlap O, then R appears alone in the body of a query Q in O and R has at least one existential variable in Q. Let R′ be the extended overlap relation for O. We show that the identity query over R is a maximally-contained rewriting of the projection query of M.R′ over attr(R).

Let Q′(attr(R)) :- M.R′(attr(M.R′)). Since R′ is an extended-overlap relation, it corresponds to an overlap O. Hence, for each query Q_i in O ($1 \le i \le |O|$) there is a relation $I.R_i$ in I where name(R_i) = name(R′) = IDB(Q_i) and attr($I.R_i$) = the duplicate-free union of the variables of Q_i. By definition of map_{MS}, there are LAV views $I.R_i$(attr($I.R_i$)) :- M.R′(attr(M.R′)) for $1 \le i \le |O|$. Moreover, these are the only LAV views that refer to M.R′.

Since R is subsumed by O, R appears alone in the body of a query Q″ in O and R has an existential variable in Q″. Since the input mapping is well-formed, that existential variable appears only in Q″. Hence, among all of the intermediate relations, only one I_j has the property that Vars(head(Q″)) ⊆ attr(I_j), namely, the intermediate relation that corresponds to Q″. Hence, by Lemma 1, I_j(attr(R)) :- M.R′(attr(M.R′)) is a maximally-contained rewriting of Q′. □

Lemma 3: (Overlap preservation) For each overlap O, there is a query Q over one relation in the mediated schema such that O is a maximal rewriting of Q.

Proof: Let M.R be the overlap relation for O. Let QO = {QO_1, …, QO_m} be the set of all overlap queries for O. We will show that Q(attr(head(O)) :- M.R(attr(R)) is the required query by showing $QO_1 \cup \ldots \cup QO_m$ is a maximal rewriting of Q. To rewrite Q w.r.t. map_{MI}, conditions (i) and (ii) of Lemma 1 apply, so by Lemma 1 $\cup_{1\le k\le m}$ (head(Q) :- I_k(attr(I_k)) is a maximal rewriting of Q w.r.t. map_{MI}. By definition of the GAV views (b2) in MSNF mapping, each I_k corresponds to a query QO_k in the overlap and hence can be replaced by body(QO_k), yielding $\cup_{1\le k\le m}$ (head(Q) :- body(QO_k), which is $QO_1 \cup \ldots \cup QO_m$ as desired. □

Lemma 4: (Extended overlap preservation) For each extended overlap query Q′, there is a query Q″ over one relation in the mediated schema whose maximal rewriting is Q′.

Proof: Suppose Q′ is the extended overlap query of query Q in some overlap O. Let M.R be the overlap relation for O and let Q″(Vars(head(Q′))) :- M.R(attr(R)) be the required query over M. We first rewrite Q″ w.r.t. map_{MI}. Since Q has an existential variable, there is a unique relation I_i in I that satisfies conditions (i) and (ii) of Lemma 1. Hence, by Lemma 1 head(Q″) :- I_i(attr(I_i)) is a maximal rewriting of Q w.r.t. map_{MI}. Since I_i corresponds only to Q, by definition of the GAV views (b2) in MSNF mapping, I_i(attr(I_i)) :- body(Q) is the only query in map_{IS} with IDB I_i. Unfolding that query into head(Q″) :- I_i(attr(I_i)) we get head(Q″) :- body(Q) as the maximal rewriting of Q as desired. □

Lemma 5: (Normalization) For each mediated schema relation R that corresponds to an overlap specification O, every attribute of R appears in the head or a body of O.

Proof: Follows directly from the definition of overlap relation. □

Lemma 6: (Minimality) The mediated schema is minimal with respect to mediated schema criteria (i) – (iv).

Proof: Partition M into the set of overlap relations M_O and the set of completeness relations M_S. We show that if any relation or attribute is deleted from M_O or M_S then M does not satisfy one of the mediated schema criteria (i) – (iv).

M_O: To satisfy overlap preservation, there must be a relation m ∈ M for each overlap O ∈ **O**. Due to normalization, we cannot combine two relations in M_O because they are not in the same overlap. Overlap and extended overlap preservation require every overlap and extended overlap query qo ∈ O to be answered using one relation. Therefore, attrs(m) ⊇ Vars(O). MSNF defines

[1] By the construction in Section 3.2, attr(head(Q)) uses the same naming scheme as the variables of any corresponding relation R, so a renaming step is not required.

attrs(m) = Vars(O), so no attributes of m may be deleted. Hence M_O is minimal given the mediated schema criteria.

M_S: By definition of completeness relation, every $m_s \in M_S$ is not subsumed by an overlap. Suppose m_s corresponds to $s \in S$. There are two cases: (1) s is not in the body of any overlap; (2) for every overlap query qo where $m_s \in$ body(qo), either qo has no existential variables or body(qo) has more than one subgoal.

Case (1): Relation m_s is the only completeness relation that corresponds to s. There can be no overlap relation that corresponds to an overlap whose body contains s, because s is not in the body of any overlap. Hence, the only queries that relate s to M are the LAV and GAV queries for the completeness relation m_s (see definitions a1 and b1 in MSNF mapping). Hence, deleting m_s or any of its attributes from M would violate completeness.

Case (2): By completeness, at least one relation must contain attr(s). The only relations in M besides m_s that contain attr(s) are the overlap relations. Let O be the overlap with qo ∈ O, and let m_O be the overlap relation corresponding to O. We claim the projection query Q(attr(s)) :- m_o(attr(m_o)) is not a maximal rewrite of Q′(attr(s)) :- s(attr(s)). By definition of case (2), either (a) qo has no existential variables or (b) body(qo) has more than one subgoal. If (a), then by Lemma 1 Q returns the union of s and the other queries in O. If (b), then since qo is connected (see definition of overlap), Q returns the subset of s that joins with another relation in body(qo). Thus Q is not a maximal rewrite of Q. Since no overlap relation satisfies completeness for s, m_s cannot be deleted from M without violating completeness. □

4. DISCUSSION

In this section, we discuss several issues related to the choice of MSNF schemas and mappings and their effect on query processing performance and completeness.

4.1 Is MSNF Always Needed?

In Section 3.4 we proved that MSNF schemas and mappings have a number of desirable properties, characterized by the mediated schema criteria. A database designer may not require all of the criteria and hence may not want an MSNF schema and mapping in all application scenarios. Still, we believe the criteria are at least a worthwhile starting point to consider which criteria are relevant in a particular scenario. To help one decide which criteria are relevant, Sections 2 and 3 show how each criterion affects the choice of mediated schema and mappings. Moreover, as will be shown in Section 5.2, there are practical examples where every type of MSNF relation and mapping is needed.

Even when MSNF is not needed, it may be useful to develop an MSNF schema as an early step of the mediated schema creation process. This yields a complete, minimal schema with a complete set of mappings. One can then prune portions of the meditated schema that are not needed for the given application and modify the mapping accordingly. However this is just a proposal. A user study is needed to determine if such a methodology has merit.

4.2 Why GLAV Views?

Section 1 showed examples to motivate the need for GLAV views for mapping sources to a mediated schema. Now that we have the precise definition of MSNF, we can reconsider the issue in more detail. The question is, why not use GAV or LAV instead?

GAV breaks down as a mapping language for mediated schemas when there are overlapping concepts in the source schemas that have additional non-overlapping information. For schemas that adhere to MSNF, this comes up when there is an existential variable in an overlap query. For example, take the mapping in Example 3. Most concepts of Flight are in common, but Go-travel contains additional information about meals.

In MSNF, the corresponding mediated schema relation in M for Flight is M.Flight(f-num, date, time, price, meal, nonstop). A GAV mapping must provide a value for each attribute in M.Flight. However, no conjunctive query can do this for Flight unless it populates nonstop with NULL. But this is undesirable as it leads to the usual ambiguity between two interpretations of NULL, namely "irrelevant" for tuples coming from Flight vs. "missing" for tuples coming from OK-Flight that have nonstop = NULL. E.g., these two interpretations are not supported by SQL.

LAV enables us to handle this situation elegantly by expressing Flight and OK-Flight as projections of M.Flight. However, LAV too has a limitation. It cannot map a mediated schema relation to the join of relations in a source. An example of this is Go-Flight and Go-price in Example 3. The obvious LAV mapping for this example would be:

Go-flight(f, t, m) :- M.Flight(f, d, t, p, m, n)
Go-price(f, d, p) :- M.Flight(f, d, t, p, m, n)
Ok-flight(f, d, t, p) :- M.Flight(f, d, t, p, m, n)

While this would allow queries on M.Flight to access Go-flight or Go-price to answer some queries, it would not allow easy access to their join. For example, it could not answer the query All-flight-info(f, t, m, d, p) :- Flight (f, d, t, p, m, n).

Due to these limitations of LAV and GAV, MSNF uses a stylized combination of LAV and GAV mappings, which are a subset of GLAV mappings. This allows both GAV and LAV views between the source and mediated schema. The subset of GLAV required is very limited; the local views are only projections.

4.3 Query Rewriting

GLAV views require a query rewriting algorithm for answering queries using views. For conjunctive queries and views this problem is NP-Complete in the number of query subgoals [12]. Nevertheless, rewriting queries is often fast enough in practice, as was shown in [19] and as is known from the widespread use of materialized views for data warehousing. Moreover, query rewriting for our limited GLAV views is more efficient than the general case; e.g., each of our LAV views has only one subgoal. In Section 5.3, we show some experiments to provide further evidence that query rewriting is fast enough.

Some additional improvement in query rewriting performance may be attainable by replacing MSNF views by pure GAV views where possible. For example, completeness relations could be mapped using GAV views. We avoided this optimization in the definition of MSNF because the lack of symmetry complicates the proof of correctness by requiring a case analysis.

4.4 Completeness

Completeness ensures that all queries that can be asked over the source schemas can be asked over the mediated schema. Naïvely, one might expect this to require that all source relations be retained in the mediated schema. However, as we showed in

Example 5, this is not true. Some source relations can be made accessible in the mediated schema simply by adding attributes to a mediated schema relation that is needed for other reasons.

5. IMPLEMENTATION & EVALUATION

We implemented an algorithm to generate MNSF in Java. With this implementation, we tested a number of hypotheses:

- Is the expressiveness of the input mapping necessary and sufficient?
- Can this method reproduce a mediated schema that has been created independently?
- Can queries over the mediated schema be rewritten efficiently?

To test these hypotheses, we used 4 of the 5 available data sets in the Illinois Semantic Integration Archive [1]: "Courses" (5 schemas), "Inventory" (3 schemas), "Real Estate 1" (5 schemas) and "Real Estate 2" (3 schemas). All schemas in the fifth data set, "Faculty," are isomorphic, so we excluded it. All schemas are in XML that is readily translatable to relational schemas. Data is available for each source to resolve ambiguities. The source schemas have an average of 2.9 relations and each relation has an average of 11 attributes. In each experiment, we merged the schemas pair-wise to create the final mediated schema.

5.1 Implementing Mediated Schema Creation

First we investigated whether conjunctive queries are rich enough to express the common cases in mediated schema creation, and whether that richness is necessary. Conjunctive queries could express most relationships required. There were two types of relationships that semantic merge could not reproduce: concatenation of data values and arithmetic manipulation.

This experiment also showed that the power of conjunctive queries was required to fully express how the schemas related to one another. Of the 16 overlaps created in this experiment, 11 (69%) required joins, and all required the use of existential variables.

5.2 Mimicking Hand-Crafted Schemas

To evaluate whether MSNF schemas are understandable and useful, we compared them to hand-crafted mediated schemas in the "Courses" and "Real Estate 1" data sets. These given mediated schemas were created with the sources in mind, but they differed, sometimes substantially, from the input schemas. We tried to replicate these mediated schemas as closely as possible using MSNF.

For the "Courses" dataset, MSNF came very close to creating the provided mediated schema. Of the six relations in the mediated schema, our approach replicated two of them exactly. The remaining four MSNF relations were similar to the given mediated schema but they included extra attributes—at least one attribute that only appeared in one source. Still, the given mediated schema included some of the attributes that correspond to existential variables in an overlap query and that appeared in the MSNF relations. So extended overlap preservation must be considered when developing a mediated schema.

The same four mediated schema relations also required joins in the input mappings. Thus, neither GAV nor LAV would have been an appropriate choice for a mapping language between the mediated schema and source schemas (see Section 4.2).

The mediated schema provided for "Real Estate 1" differed substantially from the sources, and hence the MSNF generation algorithm had more difficulties. As in the "Courses" domain, most of the generated MSNF relations contained additional attributes. Of the 105 attributes in the given mediated schema there were 22 attributes that the MSNF generation algorithm could not create. These fell into three categories:

- Fourteen attributes had no corresponding source schema data. For example, the "Superstructure" relation in the mediated schema included an "elevator" attribute, but no source included elevator information.
- Two attributes required more complex manipulation, as in the previous experiment. For example, the given mediated schema included an attribute for "number of bathrooms" which would have required adding "half baths" and "full baths" attributes.
- The remaining six attributes required examining data values. For example, one relation included "mountain view", "city lights view", and "water view" attributes, but the sources only included "view" attributes.

Hence MSNF generation generated 83 of the 91 attributes (91%) for which there was information available in data source schemas.

5.3 Rewriting Queries

Our goal was to show that queries could be translated efficiently enough for the method to be considered; our goal was *not* to show how fast the algorithm was, so the code was un-optimized. Since unfolding GAV mappings is trivial, we only compared against LAV rewriting times. We created an MSNF version of the schema that could express as much of the mediated schema as was possible in LAV. As explained in Section 4.2, the LAV mapping is less expressive, so we limited our tests to queries that could be asked over both schemas. As shown in Figure 5-1, despite the fact that both algorithms are asymptotically the same, since the LAV views also contained only one relation, LAV was in practice much slower as more query subgoals were added. The Course and RealEstate lines show rewriting times for the schemas created using the MSNF; for queries that contain 6 subgoals, both finish in under 2 seconds on a 850 MHz single-cpu PC with 256MB of RAM.

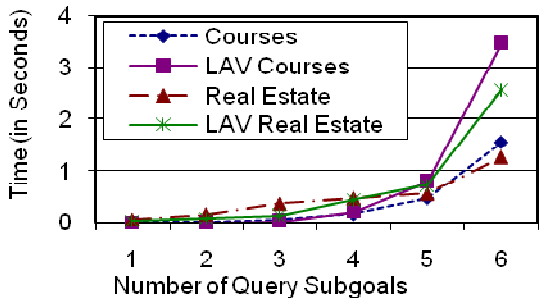

Figure 5-1 Query rewriting time is modest

To summarize the results on the practicality of MSNF: MSNF schemas are similar to hand-crafted mediated schemas. The examples required at least the expressiveness of conjunctive queries for overlap specification and of GLAV mappings for views. More expressive overlaps and views that express arithmetic and concatenation are needed. Finally, we showed that query rewriting over MSNF mediated schemas is fast.

6. RELATED WORK

6.1 Requirements

Mediated schema criteria for schema merging were introduced by Batini, Lenzerini, and Navathe in their 1986 survey of schema integration methods [2]. Their criteria are as follows:

- **Completeness and correctness:** users can get the same data from the mediated schema as they can from the sources.
- **Understandability:** the mediated schema is comprehensible to the user; i.e., queries that the user wants to ask should be easy to ask.
- **Minimality:** the mediated schema does not contain multiple ways of accessing the same concept or have any data beyond what is required for completeness and understandability.

They distinguish between view integration and data integration. View integration defines a database schema that can support a given set of views. Data integration defines a database schema to query a given set of data sources. Both require merging the input schemas into a schema that covers the input. The main difference is whether it is meaningful to define constraints between the given schemas. For view integration, the given views are views of one database. In this case, the integration process can be driven by constraints between the given views, such as referential constraints (e.g., the full-time-students view is contained in the all-students view). For data integration, the data sources are independent (e.g., student databases of different universities). In this case, there are no constraints between the given source schemas that describe how their instances are related. Rather, there are relationships that explain where the given schemas represent the same information and therefore can be collapsed.

6.2 Generating View Definitions

Among the many papers on view and data integration, to our knowledge only one other paper gives an algorithm to generate both a merged schema and view definitions (i.e., mappings) to support it, namely Melnik et al. [15]. They give formal properties that a merged schema and views must satisfy for view integration. They show that the output of their algorithm in [16] satisfies these formal properties provided that the input schemas are snowflake schemas and the input mappings are "path morphisms." A path morphism is a set of equality constraints between a pair of snowflake schemas, each of which says that a query over one input schema equals a query over the other schema. The query is limited to using join expressions that follow the snowflake relationships.

Melnik et al.'s work differs from ours in several ways: First, they address view integration, not schema integration. This has a major effect on correctness criteria. For example, in [15], they require that the input mapping between the two source schemas S_1, S_2 is equivalent to the composition of the two view definitions between the source schemas and mediated schema M. That is, mapping S_1-S_2 is equivalent to the composition of S_1-M and M-S_2. This makes sense for view integration, where S_1 and S_2 are views of the same source, but not for data integration where the instance data of S_1 and S_2 are unrelated.

Second, in [15] the inputs are limited to snowflake schemas and path morphisms. By contrast, we allow arbitrary relational schemas and conjunctive queries. And third, since their input does not characterize overlaps as conjunctive queries, they have no notion of extended overlap or extended overlap preservation in either their correctness criteria or merge algorithm.

An extension of the merge algorithm of [15] with reverse engineering support is in [7].

6.3 Conflict-driven Schema Merge

There are many papers on schema merging algorithms that produce an integrated schema from the source schemas. Since there are too many to cover here, we discuss just a few from the last decade as a representative sample. We refer the reader to the Batini et al. survey [1] for most earlier ones.

Schema merging's goal is to collapse overlapping schema elements in the output schema. This may not be straightforward due to conflicting representations in the sources. For example, Buneman, Davidson, and Kosky [4] merge two schemas by collapsing classes and attributes having the same name. If a class C appears in both input schemas S_1 and S_2, but C has two different attributes in S_1 and S_2 that have the same name but different ranges, then the naming conflict has to be solved in the merged schema. They describe a merge algorithm that solves this type of conflict and prove that it produces a unique output independent of the order in which the conflicts were resolved. In our model, such a naming conflict would show up as a violation of well-formedness of the overlap specification (see Section 2.3). We assume such conflicts are eliminated in a preliminary renaming step.

A later algorithm by Spaccapietra and Parent treats a broader class of conflicts [22]. Unlike [4], they assume an explicit set of correspondences between the input schema elements. They describe merge procedures as a sequence of integration rules: integrate objects, integrate links, integrate paths, etc. They create correspondences between the merged schema and input schemas, but do not show how to turn those correspondences into views.

We followed a similar approach in a merge algorithm we presented in a previous paper [20]. We allowed more expressive input mappings than [1], where the mapping itself was a schema, and resolved other conflict types in addition to those of [4] and [1]. Like [1], [20] generates correspondences but not views between the merged schema and input schemas. In fact, the present paper started as an attempt to extend [20] to generate views, which led us to the more general question of choosing correctness criteria and proving the merged schema and views satisfy them. The algorithm in [20] can simulate the algorithm in Section 3.3 of this paper by interpreting the output correspondences as view definitions. Some details are in [18].

Both-As-View (BAV) focuses on updating a mediated schema based on the integration of new source schemas [14]. A BAV mapping calls for adding, deleting, and renaming attributes and relations in the mediated schema. Our work differs from theirs in several ways. First, our work creates the mapping between the mediated schema and the sources. Second, their method does not guarantee that the resulting mediated schema adheres to many of our mediated schema criteria. Finally, our work creates the mediated schema based on the relationship of the sources to each other, not to a previously existing mediated schema.

6.4 Constraint-Driven Merge

Two early works, by Casanova and Vidal [6] and by Biskup and Convent [3], study view integration using constraints between two

given view schemas. In both papers the goal is to create a merged schema that is complete, reduces redundancy, and is minimal.

In [6], Casanova and Vidal assume an input consisting of view schemas that are in Boyce-Codd Normal Form and have constraints within and between views: key constraints, inclusion (i.e., foreign key) constraints, exclusion constraints (the key values in two relations are disjoint.), and union-key constraints (given attribute A in relations R_1 and R_2 with common key K, $\pi_K(R_1) = \pi_K(R_2)$ implies $\pi_A(R_1) = \pi_A(R_2)$, where $\pi_K(R)$ is the projection of R on attributes K). To create the merged schema, they start with the disjoint union of the view schemas and then apply optimizations to reduce the number of relations and columns based on the constraints. For example, union-key constraints may indicate that a set of relations can be replaced by one relation. They do not produce view definitions (i.e., mappings).

In [3], Biskup and Convent extend [6] by using a formal notion of completeness that is essentially Hull's query dominance [9]. Their integration constraints between views are inclusions, exclusions, identities (i.e., bidirectional inclusions), and selection constraints (identities on a subset of tuples satisfying a selection condition). They optimize to reduce the size of the schema by applying constraints between views. Relations connected by exclusion, identity and selection constraints are combined, provided that this does not break any constraints within views. The algorithm defines the output merged schema, but there is only an example of a mapping between the view schema and global schema, not an algorithm for generating it. However, the output mappings between the views and the global schema are essentially the same as the input mappings for most of the given constraints, so an algorithm to cover these cases would be straightforward.

7. CONCLUSION AND FUTURE WORK

We have presented the first algorithm for generating a mediated schema and view definitions for data integration from a given set of source schemas and specifications of overlapping information. We defined a new normal form for a mediated schema and its view definitions, called Mediated Schema Normal Form (MSNF). We developed formal correctness criteria for a mediated schema and view definitions and proved that an MSNF mediated schema and view definitions satisfy the criteria. We presented an algorithm that generates an MSNF schema and views and described what we learned from its implementation.

We see several opportunities for future work. First, it would be useful to enrich the mapping language beyond conjunctive queries, include functions and constraints, especially key and foreign key constraints. Second, we would like to incorporate our algorithm into a tool for developing integrated schemas, so that a user study could evaluate its utility. Third, we feel that enough new work has appeared since Batini et al.'s survey [1] that a new survey of schema integration techniques would be worthwhile.

8. Acknowledgements

We thank Xun Sun for helping with the experiments in Section 5.

9. REFERENCES

[1] C. Batini, M. Lenzerini, and S. B. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Computing Surveys* 18(4), pp. 323-364, 1986.

[2] P. A. Bernstein, "Applying Model Management to Classical Meta Data Problems," *CIDR 2003*, pp. 209-220.

[3] J. Biskup and B. Convent, "A formal view integration method," *SIGMOD 1986*, pp. 398-407.

[4] P. Buneman, S. B. Davidson, and A. Kosky, "Theoretical Aspects of Schema Merging," *EDBT 1992*, pp. 152-167.

[5] A. Cali, D. Calvanese, G. D. Giacomo, and M. Lenzerini, "On the Expressive Power of Data Integration Systems," *ER 2002*, pp. 338-350.

[6] M. A. Cassanova and V.M.P. Vidal, "Towards a Sound View Integration Methodology," *PODS 1983*, pp. 36-47.

[7] M. Gubanov, P.A. Bernstein, M. Moshchuk, "Model Management Engine for Data Integration with Reverse-Engineering Support," ICDE 2008, to appear.

[8] A. Y. Halevy, "Answering Queries Using Views: A Survey," *VLDB J. 10(4),* pp. 270-294, 2001.

[9] R. Hull, "Relative Information Capacity of Simple Relational Database Schemata," *SIAM J. Comput.* 15(3): 856-886,1986.

[10] Illinois Semantic Integration Archive. http://pages.cs.wisc.edu/~anhai/wisc-si-archive/.

[11] M. Lenzerini, "Data Integration: A Theoretical Perspective," PODS 2002, pp. 233-246.

[12] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava, "Answering Queries Using Views," *PODS 1995*, pp. 95-104.

[13] A. Y. Levy, A. Rajaraman, and J. J. Ordille, "Querying Heterogeneous Information Sources Using Source Descriptions," *VLDB 1996*, pp. 251-262.

[14] P. McBrien, A. Poulovassilis, "Data Integration by Bi-Directional Schema Transformation Rules," *ICDE 2003*, 227-238.

[15] S. Melnik, P. A. Bernstein, A. Halevy, and E. Rahm, "Supporting Executable Mappings in Model Management," *SIGMOD 2005*, pp. 167-178.

[16] S. Melnik, E. Rahm, and P. A. Bernstein, "Rondo: A Programming Platform for Generic Model Management," *SIGMOD 2003*, pp. 193-204.

[17] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan, "The Use of Information Capacity in Schema Integration and Translation," *VLDB 1993*, pp. 120-133.

[18] R. Pottinger, *Processing Queries and Merging Schemas in Support of Data Integration*, PhD thesis, Univ. of Washington, 2004, http://www.cs.ubc.ca/~rap/publications/thesis.pdf

[19] R. Pottinger and A. Levy, "A Scalable Algorithm for Answering Queries Using Views," *VLDB 2000*, pp. 484-495.

[20] R. A. Pottinger and P. A. Bernstein, "Merging Models Based on Given Correspondences," *VLDB 2003*, pp. 862-873.

[21] R. A. Pottinger and A. Y. Halevy, "MiniCon: A scalable algorithm for answering queries using views," *VLDB J.10 (2-3),* pp. 182-198, 2001.

[22] S. Spaccapietra and C. Parent, "View Integration: A Step Forward in Solving Structural Conflicts," *TKDE 6(2)*, pp. 258-274, 1994.