# Scalability Issues in Data Integration[1]

**Arnon Rosenthal, Ph.D.**
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
(781) 271-7577
arnie@mitre.org

**Len Seligman, Ph.D.**
The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102
(703) 983-5975
seligman@mitre.org

**Abstract**

*Data integration efforts often aim to give users access to multiple data sources through queries (and other requests) against a global schema. As sources change, new ones become available, and others become unavailable (at least temporarily), it becomes very burdensome to maintain the necessary mappings and other metadata. We compare the administrative labor and data accessibility for two popular approaches: federated databases that derive each table in the global schema as a view over sources, and source-profile systems that describe each source's offerings as a view over the global tables. We then propose a hybrid process that combines their advantages.*

Key words – Data integration, data administration, scalability, federated databases

## 1. Introduction

*Data integration* is the ability to meaningfully exchange information among separately developed information systems. (In [Sel01], we provide a detailed discussion of the steps required to achieve data integration and XML's contribution.) The problem is ubiquitous in all large organizations. Large-scale data integration is currently quite costly both for initial implementation and for continued maintenance. The Department of Defense had disappointing results from major initiatives in the early 1990s (e.g., DOD Standard 8320.1, which standardized thousands of data elements, with little connection to implemented systems). We discussed barriers to success, and the need for incentives, in [Ros00]. Large organizations have generally achieved partial integration, and are constantly facing new requirements and new external partners. Consequently, we propose that the right goal is to accumulate knowledge and tools that provide partial integration for current needs, and that support both further integration (as needed) and change.

Two critical steps in providing data integration are:

- developing and maintaining a schema for the integrated environment, here called the *global schema* or *mediated schema*
- creating (and maintaining) the mappings between the individual data sources and the mediated schema

There are a variety of approaches to developing the mediated schema [Bat92] including top-down, bottom-up, and hybrid strategies.

This paper focuses on the second problem, creating and maintaining the mappings. To support integrated access, these mappings must be executable—e.g., SQL views. For large-scale integration efforts, this view creation step is currently quite costly and labor intensive. Our goal is to explain the nature and tradeoffs for the major alternative models for this step. Finally, we propose a new alternative, which merges aspects of the two major approaches.

---

[1] Presented at *AFCEA Federal Database Colloquium*, San Diego, 2001.

For concreteness, we discuss a particular goal – enabling queries against a mediated schema that captures the domain of interest for all the user requests, and is used as the center of all administration. Source(s) will be (somehow) described relative to the mediated schema. User requests are expressed as queries against the mediated schema (or against views built above it). The mediator system takes such a request and finds a way to execute it, using capabilities of the sources plus its own DBMS-like capabilities. Systems like this are becoming increasingly common; one example is the Global Command and Control System (GCCS) Common Operational Picture - Combat Support Enhanced (COP-CSE).

This paper is a mid-June snapshot of work in progress. A more detailed version will be made available at *http://www.mitre.org/resources/centers/it/staffpages/arnie/*.

### 1.1 Scope

The essential data administration task of government organizations is to describe their world, i.e., their data sources, the interfaces they wish to support, and how the two interrelate. These are the aspects that differentiate an army's data from a soft drink manufacturer's or a tax authority. *Generic* capabilities can come from vendors; in particular, we expect continual improvement in metadata managers and query engines. In fact, we scope our problem by assuming vendors will provide metadata management tools equal in power to a hyper-caffeinated database PhD who has no knowledge of your organization's mission, beyond what is provided in structured metadata. This person understands data models and query languages, and responds quickly 7 x 24, but does not understand the meaning of database table and attribute names and is not shown any English documentation.

Similarly, we assume that the system has a very clever distributed query processor (automated or human) that handles queries in a formal query language such as SQL or XQuery. It does an excellent job of exploiting schema, constraints, operators' algebraic identities, and statistics metadata to find an execution strategy. The techniques apply also to mediating user-requested updates and to push (subscribe-to-changes). We will occasionally remark on these, but will not really address them.)

Finally, creation of a single central schema can be difficult for moderate domains; for a huge domain (e.g., the Department of Defense) global agreement is effectively impossible [Ros97]. Nevertheless, we bound our problem by excluding this issue. We believe that the insights concerning a single mediated schema will provide a starting point for the even larger problem.

Data administrators must maintain sufficient metadata so the clueless technologist or tools can perform mediation. This paper's main goal is to clarify: For integration of large numbers of diverse tables, how easy will it be to set up the metadata initially, and to maintain it as the system evolves?

### 1.2 Terminology

We use relational terminology, with the understanding that the ideas are 90% independent of the data model formalism (e.g., they apply to XML or object views). Also, when we refer to "SQL views," they may include foreign functions that do domain-specific computations (e.g., interpolation, or converting JPEG to GIF), as in SQL:1999.

*Scalability* refers here to the *data administration* difficulties in creating and maintaining large systems (not to run-time performance): How much do you need to know to add a source, or delete one? Can one look at just the mediated schema and the source? Perhaps add just one or two neighbors for each table? Or does one have a giant view that performs a join across 30 different ForeignTaxAuthority sources? Another important dimension is how much does one need to read and change if there is a small change, e.g., to a single attribute. Does the model allow you to edit just that attribute and how it relates to its federated version, or does one need to edit that same 30-table join?

## 2. Existing Approaches for Creating and Managing Mappings

This section describes and compares the two major approaches to in the literature (and to some degree in products): *federated views* and *source profiles-driven.* In both approaches, the mediated

schema represents a virtual database. The term *mediated schema table (M-table)* denotes a table specified in the mediated schema (*M-schema*).

## 2.1 Federated views (Global as View)

In the federated approach, a system integrator creates an integrated view (the M-schema) based on a set of large, complex queries, each of which transforms and combines information from multiple sources. The approach is called *Global as View (GAV)*, because each table in the "global" M-schema is an SQL view.

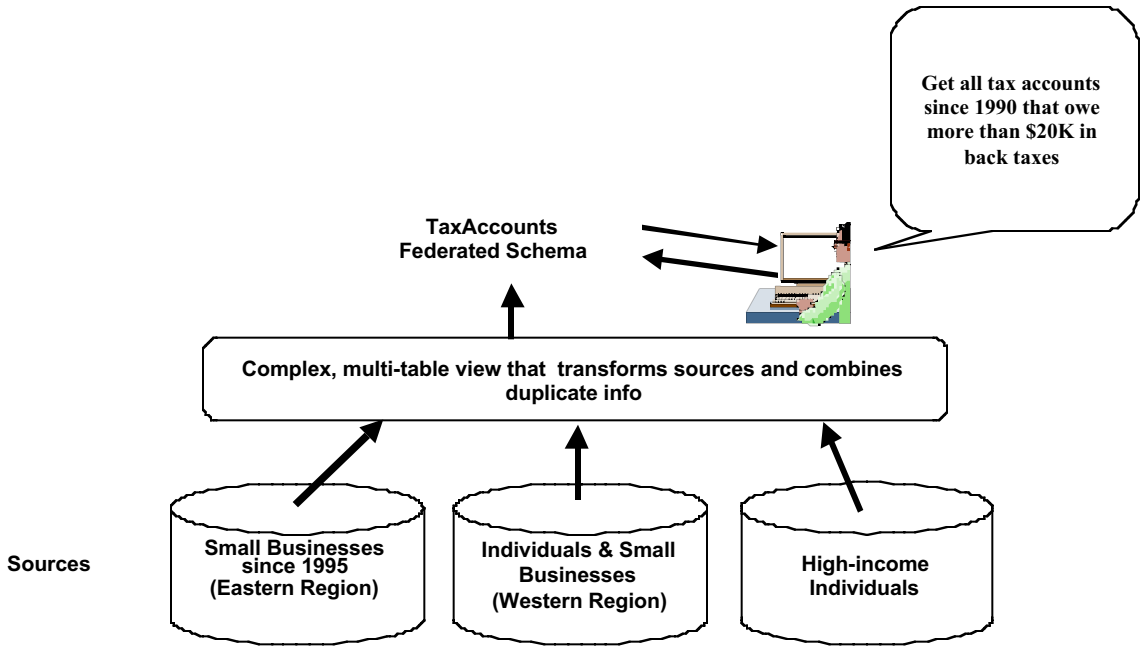


Figure 1: Tax Administration example with Global as View

Figure 1 illustrates the GAV approach with a simplified tax administration example. Historically, regional processing centers developed their own systems. Current modernization efforts are producing a functional rather than geographic viewpoint. Suppose the mediated schema supports collections on overdue accounts. The mediated schema includes a TaxAccount M-table with attributes TaxID, Name, Year, Balance, TotalDue, and TaxableIncome. Suppose also that this mediated schema exists to support analysis of collections on accounts that owed large amounts. In this example, TaxAccount is a virtual table that is populated via a multi-table SQL view. The view transforms and combines data from a small business accounts table at the Eastern Region, a combined individuals and small business accounts table at the Western Region, and a table of high-income individual accounts at Headquarters. Users issue queries against the mediated schema—e.g., "get all tax accounts that owed more than $20K in back taxes in some year since 1990." A federation query processor transforms the query into subqueries that are executed at each site and transforms, combines, and filters the results according to the view and query conditions, and returns the result to the user.

GAV is well understood, and a natural choice for the short term. Researchers have studied it extensively since the mid-1980s [Osz99]. Today's query processors can handle queries over the global views in distributed relational DBMSs; tomorrow's will handle web sources (e.g. presented as lookup functions rather than tables) and will automatically create execution cost estimators for foreign sources. For administration, we expect good GAV schema integration tools to emerge from current research prototypes (e.g., [Mil01] at IBM).

However, GAV suffers from two serious shortcomings. First, administration of GAV scales poorly, because GAV requires that administrators write and maintain views that reference many sources. Adding, deleting, or changing any of these sources requires inspecting a large view. If one has 2000 attributes and 1% change per month, then one has one change every working day, and each change

may affect multiple views [Goh99]. The toughest part, for scalability, concerns knowledge of pairs of sources, e.g., which is more accurate and how one can match (say) individual taxpayer identities (when TaxID numbers might be lacking or incorrect) across them.

Second, GAV provides no way for users to judge the completeness of results that are returned. In the example, the user has requested tax accounts since 1990. She gets back a "bunch of accounts" which satisfy the criteria. However there might be some missing data. There is nothing in the SQL view that populates TaxAccount that tells the user a potentially important piece of information: the Eastern Region data source only has accounts since 1995.

### 2.2 Source profiles (Local as View)

An alternative approach that has received much recent attention from researchers is to provide *source profiles*. The mediated schema is purely a domain model; it is seen as describing a *conceptual database* comprising all the world's information on each type. A source profile describes which portion of the world's information the source has. A new type of query processor matches a user request against the currently available sources, and devises a query that computes the desired result. The approach is often called profiling or Local As View (*LAV)*; when we use it as a component of a larger mediation approach, we will sometimes refer to it as *downward view* (from M-schema to sources). Figure 2 illustrates a LAV treatment of the tax administration example.

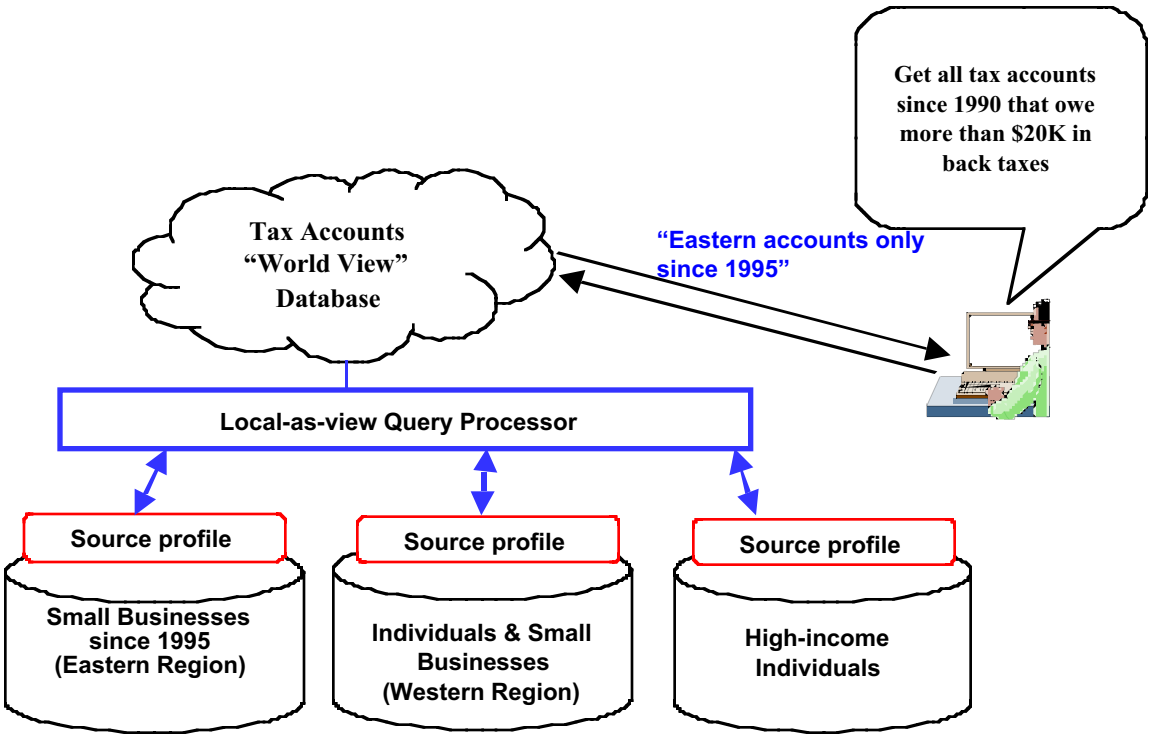


Figure 2: Local as View example

LAV's great advantage is in scalability of administration: each source profile is independent of the others. As result, when a new source is added or removed, or a source database schema (or its population) changes, administrators edit only the relevant source profile without having to change any of the others. The approach is also very tolerant of additions to the mediated schema– the profile of a given source remains sound though it may no longer be complete. All previous capabilities are intact; however, the system does not yet know if that source contains information relevant to the additions.

A second advantage is that LAV is explicit about the coverage of each source. In the example in Figure 2, it would be apparent from the source description for the Eastern Region that the results

from that source only go back to 1995. This information can be used to avoid unnecessary access, and to inform the user if the available sources did not cover all instances of interest. One could add this scope information as constraints on GAV sources, but query processors may not exploit the constraints, and (worse, from our point of view) one then is doing duplicate administration, i.e., producing the LAV profiles to derive sources *plus* GAV views to derive mediated schema tables. The feedback can be further improved if one documents for each source whether its content is considered *sound* (i.e., each value in it is correct) and *complete* (possessing all relevant instances). If a query without negation uses only sound sources, then its result is sound. If it appears to have complete coverage and uses only complete sources, then its result is complete.

Yet LAV is not ready to be a full replacement for GAV. Current DBMSs do not support LAV (though query processing over materialized views gives some of the necessary mechanisms). More fundamentally, by delegating the match process to the query processor rather than requiring administrator input, LAV removes opportunities for detailed control over how tuples from multiple sources are merged to produce a result tuple.

The best solution seems to be a hybrid: LAV to understand source contents and (implicitly) specify a simple way to combine the sources, followed by GAV to give details of the combination. The next subsection explores some details of LAV and GAV that will be needed for our hybrid process description.

## 3. Suggested Approach

We now describe an approach to creating the view definitions needed for data integration. We take well known techniques and organize them to be 1) more scalable and 2) achievable by domain experts rather than technologists. The approach decomposes view creation into small metadata-capture steps, and generates SQL from that metadata.

A main theme of recent computer science is to replace programming by *"describe and generate"*. For example, relational DBMSs replaced programming of access strategies by generating them based on the user's desired result plus descriptions of the database statistics and physical structure. Analogously, early federated approaches required administrators to write SQL code to produce the federated view. More recent systems ask the user a series of narrow questions, and then generate the SQL code [Yan01].

The goal is to shift work to domain experts who know the data but have minimal programming or DBMS skills. These domain experts provide descriptions via menus and other GUI constructs. When change occurs, one can edit a small chunk of metadata. Beyond this, integration may require real programming to transform attribute representations (e.g., from JPEG to GIF), but such efforts are reusable and localized (i.e., require neither DBMS expertise nor knowledge of the whole schema). The mediator has such knowledge, and can insert appropriate function calls into SQL expressions.

The approach ameliorates the weaknesses of GAV and LAV. Compared to GAV, it avoids writing and maintaining large view queries over multiple sources. It also naturally captures the scope of each source; this allows the query processor to eliminate irrelevant sources and give hints about the result's completeness. At the same time, unlike LAV, the approach allows administrators to specify how M-table tuples shall be produced out of tuples from different sources.

At each step, administrators capture enough metadata to generate upward views that simplify the picture beneath. This metadata is kept, and if the system changes, one edits discrete items of metadata, rather than a large query language statement. We describe the view tables produced at each stage, but one would generate the view only if an administrator wished to see it.

*Step 1: Preliminaries:*

*a. Domain experts familiar with the global objectives express the domain model users will see as an external M-schema.* (The external schema is presented to users and their applications, while an internal schema provides additional attributes—described below—needed for administration.)

*b. Domain experts familiar with a source system match attributes between sources and the M-schema, and describe the representation for each attribute.* This knowledge is captured as metadata

on source attributes (or attribute groups, e.g., [Day, Month, Year]). They also provide predicates that restrict each source table to match the desired coverage of the M-table (e.g., deriving TaxableBusiness from Organization).
*c. Vendors and programmers familiar with the domain datatypes provide a library of transformations for attributes* (e.g., Miles to Kilometers, JPEG to GIF) and attribute groups (e.g., Day/Month/Year to DateString).

The mediator then automatically identifies further attributes on which the external M-schema attributes depend. It creates an *internal M-schema* that includes them, as explained in the Section 4. The mediator can now generate upward views showing the data contributed by each source table, with relevant single-table metadata for each attribute.

*Step 2: Describe Each Source's Coverage (w.r.t. the M-schema)*

Both query processors and end users want to know what data each source provides. An expert familiar with each source system expresses this as a query that derives that source from the internal M-schema. The expert can also indicate if the source is sound and/or complete. The above are all done one source at a time (LAV-style), so administration scales well. Also, if one knows that one source table (or view of it) contains all the tuples in a second source, this *inclusion* should be documented. The phenomenon usually occurs when the first source provided the data for the second.

The system now generates an upward view that *accumulates all source information relevant to each M-Table, without attempting to join or resolve inconsistencies.* Technically, the accumulation is an *outerunion*; where sources differ on what attributes they provide, one pads with nulls. We assume also that the system generates an attribute that indicates the source that provided each tuple, for late use.

For many types of profiles, each source is a subset of a projection of an M-Table. If this is so, then at this point, the mediator can generate an upward view that derives a population for each M-Table. For example, the same taxpayer may have made payments in both regions, and have been high income during 2000; a tuple is created for each source tuple. Steps 3 and 4, the least scalable, gather information about the same real world entity.

*Step 3. Determine which tuples represent the same real-world entity.*

The knowledge for this step begins with declared foreign keys (generally between tables in a source schema). Domain experts (aided by tools) match attributes across source tables that contribute to the same M-table (a narrower universe than all source tables). They may specify that an external system (e.g., a Name/Address-matching expert) will carry out the match predicate. They may also specify a predicate to identify alternate (non-identical) reports within a table, e.g., two Payments with the same social security number but non-identical names (Jane Doe and Jane Smith).

Next, one asks a domain expert questions such as "If a Taxpayer has no matching Payment tuple, should it be included or excluded". Guided by such descriptive information, the mediator software creates a join, left outerjoin, right outerjoin, or full outerjoin (shielding the domain expert from the technical definitions).

At this point, the mediator can produce an upward view that contains a single M-table tuple for each entity in the "real world" of the integrated schema. When there are multiple tuples with the same entity key, the view produces a repeating group for each attribute that has multiple values. For example one taxpayer's entity might be

SS#, {(name1,source1), (name2,source2)}, {addr1, source1)(addr2, source2), (addr3, source3)}

Section 4 contains further explanation.

*Step 4. Determine what to return if data values are multiple*

Domain experts specify what is to be returned from each repeating group. Mediators will include tools that provide multiple options, for nonprogrammers to choose, plus an exit for user-supplied code. For example, a taxpayer who changed residence might have payments in both Eastern and Western regions. For Address, the expert might choose to use the address from the source with the

most recent payment; for name, the choice might be to return all names with separator string "*aka*" (also known as).

## 4. Discussion of the Proposed Process

Now that the process is described, here is the place to discuss outstanding issues with each step (Section 4.1), scalability (Section 4.2), and semantics of profile-driven systems (Section 4.3).

### 4.1 Discussion of Individual Steps

**Step1 discussion:** Attribute matching is difficult, but has an extensive literature, surveyed in [Rah01], and we expect practical tools to emerge soon. For representations, one just captures descriptions of how each source represents each attribute. Ideally, the user query (or user profile) will indicate the desired representation for each attribute in the result; the mediator will automatically insert calls to the appropriate transformations [Goh99]. We will not discuss this issue further.

The new issue is the need to augment the M-Schema to make it more robust, in two areas: fine-granularity data and linking attributes. The mediator does this automatically, based on attribute and entity matches from 1b. For fine granularity, suppose there is an M-table column of WeeklyRevenue for each zip code, but the Florida source reports DailyRevenue since 1999. Clearly this source has usable data, but it cannot be described as a view over the M-table. A simple solution is to add DailyRevenue to the internal M-table; and inform the query processor how WeeklyRevenue can be derived from it. (By doing it this way, we can capture the fine-grained Florida data., while retaining information on the source's completeness, costs, and so forth).

Link information provides a subtler reason to extend the internal M-schema. For example, suppose an M-table tracking sales has attributes (ItemType, Price, DeliveryAddress), but one source divides the information as Sale(ItemType, Price, Order#) and ShipUPS(Order#, DeliveryAddress) in another table. We can straightforwardly derive portions of each source table (with Order# omitted) from the M-table, but that information will be insufficient to tell us which item to ship to each address. One cannot reconstruct the M-table from its projections. By placing Order# into the M-table, we preserve the connecting data. (In a GAV system, an administrator could simply define the join view. But that approach scales poorly in an environment with deliveries spread among slightly different tables for ShipFedEx, ShipBNrail, …). In richer examples, one may add linking entities rather than just attributes. For example, some participants might key on Shipment# rather than Order#, and one might need a table Order(Order#, Shipment#).

To accommodate schema augmentation for the above purposes, and also when new sources with interesting data are added, we employ two layers: sources are mapped to the internal M-schema, which may be extended freely. The queries that derive tables of the external M-schema can be generated automatically by comparing attribute names and employing metadata about roll-up and (at step 3) joinability. If there are user communities with different requirements, one may create multiple external M-schemas.

**Step 2 discussion:** Source profiles are important for performance; they let the query processor identify sources that cannot be relevant to the user's query, and need not be searched. In addition, if there are inclusion constraints among the sources (e.g., one source was derived from another so the data is identical, or a subset), these are declared. That will enable the mediator to omit redundant or irrelevant sources. For example, our first source description profile has (Date>1995); this source is irrelevant to a user query requesting (Date < 1990). When data is partitioned, sometimes only one of a large set of sources is relevant, so the time saving can be large.

Inclusion assertions can also be helpful. (Strictly speaking, we mean subsumption rather than inclusion, i.e., that a subset of the attributes of one tuple contains the other). If a more thorough and equally sound source is available, the query processor can skip the inferior one. (Unlike the other metadata at steps 1-2, inclusion assertions involve multiple sources. However, unless deduced from unary soundness and completeness assertions, inclusions tend to arise when there is a data flow

between the systems. Such flows deserve to be part of the system model, if only to determine whether we really want to maintain the redundant sources. Also, eliminating them from query results helps avoid spurious evidence when choosing among alternative values (at step 4), when several sources agree on a value, but really have just copied the same report.

Techniques at this step to minimize the number of source tables accessed are particularly vital in Web environments, where the number of sources can be enormous. It is valuable to eliminate unpromising sources early.

**Step 3 discussion:** We may have many tuples that purport to describe the same real world entity. This step concentrates on grouping data for each entity, i.e., its world is reported data rather than external reality (which comes in at step 4). (Step 2 has eliminated extra reports due to source inclusion; these would provide only redundant values.)

One may wish to defer view generation over multiple sources (steps 3,4) until one knows what sources are available at run time. (Also, the end user may wish to control the heuristic matching processes there.)

The relational model offers limited support, but both SQL99 (in the Object facilities) and XML allow repeating groups. A display with repeating groups like the one shown is far preferable to having a flat representation that takes a Cartesian product of repeating groups (six separate tuples here).

**Step 4 discussion:** Value resolution is not offered in DBMSs (except for query language constructs like "coalesce" that make rules easier to write and optimize). As "data cleaning", this area is beginning to receive attention from vendors and researchers. The first generation of products, aimed at data warehouses, helps an administrator specify rules. Researchers are examining use of integrity constraints and machine learning, plus ways to make the result depend on client preferences (e.g., can the client accept a list of alternatives). It is sometimes desirable to drill down into sources for further hints, e.g., the date of the payment in the example with step 4.

Some techniques work by attaching metadata describing source preferences or reliability, e.g., that the current user considers source2 to be authoritative, in case of conflict. As discussed below, these techniques differ in scalability.

### 4.2 Scalability

Our process is scalable for several reasons. First, one can edit the descriptions provided at each step, separately. There is no effect on descriptions from earlier steps, and often none on later descriptions. Automated generation can then proceed. Research is needed into algorithms that determine which descriptions are invalidated, help administrators update the information, and then regenerate the affected view information. Second, use of LAV at the initial stage partitions the problem by sources. (At steps combining across sources, this is somewhat less effective, but as noted for step 3, one still has a smaller universe, i.e., only sources relevant to a particular M-Table).

We expect that derivations of M-tables from sources will typically be more difficult to write than derivation of sources from M-tables. We expect that #source tables > # M-tables, so on average, more source tables will be involved in deriving an M-table than vice versa. Also, when multiple source tables are related to one M-table, they will often be from different organizations and there will be no administrator familiar with all of them. The M-schema is easier to browse than "all possibly relevant tables in all other sources".

However, only a portion of the metadata is captured at the LAV stage. At later stages, we try to separate the metadata that is relatively easy from the tougher items, and to let each be maintained separately. The remainder of this section proposes a simple categorization to estimate the difficulty of metadata acquisition.

Metadata that describes a single source table (here called *unary metadata*) is relatively easy to maintain. The amount of new metadata needed to introduce a new source is constant, independent of the number of existing sources. Each source table probably has a knowledgeable administrator; cross-source knowledge is rarer. We have arranged our process to maximize the amount of metadata that is unary, as discussed in Section 2.2. Unary metadata includes the view that derives the source table

from M-tables, and assertions of soundness and completeness. For example a table of US States and their postal codes is likely to be sound and complete; a column describing taxpayer addresses might be complete but not sound (due to recent changes). Finally, one source might be declared authoritative for tax rate information, to be used in preference to all others should there be a conflict.

A second category, here called *pseudo-unary* metadata, describes a single source, relative to some globally motivated scale, e.g., accuracy, responsiveness, or some aggregate desirability. It is often difficult to get agreement on how to measure these, and sources may have reasons to rate themselves overoptimistically. Machine learning techniques might be helpful in moving difficult problems (e.g., value conflict) from the tough category below, up to here.

Finally, sometimes one needs to look explicitly at all the pairwise metadata relationships among values being combined. If adding a source requires assessing its credibility against all existing sources, this will lead to high costs as the system scales up. Fortunately, one needs to compare only against related sources, and the necessary information is already being captured (e.g., foreign key, is derived from, same semantics but older version, etc.).

### 4.3 Semantics of Queries Expressed as Profiles

The semantics of queries over overlapping, unsound sources are quite difficult. We understand what portion of the conceptual world each database approximates, but the combination effect and the omissions are hard to explain.

One way to see it is that the conceptual database has some unknown population, and each source presents a view over the conceptual database. Now even if the sources are sound and complete for their advertised coverage, they may not present enough information to infer the conceptual values. For example, if no source includes DateOfBirth in its scope, then the sources provide no information about that part of the conceptual database. That is, the conceptual database can contain any arbitrary values for DateOfBirth and still be consistent with the sources [Gra99]. A further semantic difficulty with LAV is that the profiles give no hint about how to resolve conflicts – that is why the process used upward views for steps 3 and 4.

Perhaps the best way to understand the semantics is that the query processor in effect splits the user query into two parts, $Q_{user} = Q_{sources} \cup Q_{residue}$, and returns only $Q_{sources}$. Its goal is to find the largest possible set $Q_{sources}$ that is contained in $Q_{user}$(conceptual db). The source derivation predicates make it possible to express $Q_{sources}$ and $Q_{residue}$ in terms of the Mschema. However, there are two severe difficulties. First, pragmatically, with today's query rewrite technology [Hal01], the resulting SQL expressions convey little intuition about the result. Second, the decomposition is not uniquely defined. We omit further details.

## 5. Conclusion

Organizations need not choose between federation and source profiles -- both viewpoints are part of the large scale solution. In both approaches, users' requests take the form of SQL queries over the mediated schema. The mediator uses its knowledge of sources to determine what query over the sources generates the proper result. The two approaches differ in the knowledge available to the mediator, and hence in how the mediator constructs the query.

In seeking scalability, we recommend capturing knowledge as reusable, editable metadata rather than as large SQL views. One achieves much better locality, the knowledge is available should circumstances change, and one can have domain experts do more of the work. Unary metadata is particularly valuable.

Some parts of the process are fairly easy with today's products, or with simple scripts. Once matching is done, humans can identify relevant sources, and rapidly construct the upward view of step 2. Products offer some help in the data connection and cleansing of Steps 3 and 4. Humans can reason about soundness and completeness, and provide that metadata on M-tables (relative to sources' coverage). We believe (but have *not* demonstrated) that a process like ours can, even with today's tools, reduce costs and improve flexibility.

We described the process as if one had brilliant mediators and query processors. Today's systems do not approach that standard, but human experts can substitute (at higher cost and with less immediate response, of course). To *greatly* reduce administration costs, more comprehensive tools (but little new research) are needed.

There are also important capabilities that users currently lack in today's systems, even after manual integration efforts have produced upward federated views. Further research seems needed, though, before one can provide them. First, for domain experts who are not technical wizards, we would like to generate an *understandable* explanation of the coverage a query has obtained from available sources, including an understandable explanation of the residue. Second, one wants to give an understandable explanation of the lineage (i.e., sources and computation history) of the result the user receives. Third, one wants to provide fault tolerance, and thus need fully automatic regeneration when source availability changes. (One also needs a principled, automated way to change request semantics, e.g., to ignore some predicates if necessary data is inaccessible.) Fourth, query processors need to work better with enormous numbers of potential sources.

Finally, one must consider the important parts of data integration semantics that were beyond our scope. Techniques for identifying matching types of information are surveyed in [Rah01]. Approaches to combining multiple heuristics for matching are described in [Doa01, Mad01]. Ways of helping the user provide descriptions are explored in [Yan01].

Our next research goals are to understand which parts government organizations can do immediately, and to provide a more detailed adaptation of our process to doing incremental maintenance.

## References


[Bat92] C. Batini, C. Ceri, S. Navathe, *Conceptual Database Design,* Benjamin/Cummings, 1992.

[Doa01] A. Doan, P. Domingos, A. Halevy, "Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach" *ACM SIGMOD Conf.*, Santa Barbara, 2001, pp. 509-520.

[Goh99] C. Goh, S. Bressan, S. Madnick M. Siegel "Context interchange: new features and formalisms for the intelligent integration of information", *ACM TOIS,* Vol. 17(3) 1999.

[Gra99] G. Grahne, A. Mendelzon, "Tableau techniques for querying information sources through global schemas", *Proceedings International Conf. on Database Theory ( ICDT), pp. 332-347, 1999*

[Hal01] A. Halevy," Answering Queries Using Views: A Survey", *VLDB Journal,* to appear.

[Mad01] J. Madhavan, P. Bernstein, E. Rahm , "Generic Schema Matching with Cupid", to appear Very Large Data Base Conf., 2001.

[Mil01] R. Miller, M. Hernandez, L. Haas, L. Yan, C. Ho, R. Fagin, L. Popa, "Clio: A Semi-Automatic Tool For Schema Mapping", *ACM SIGMOD Record, web edition,* March 2001. http://www.acm.org/sigmod/record/issues/0103/index.html

[Osz99] M. T. Ozsu, P. Valduriez, *Principles of Distributed Database Systems*, 2/e, Prentice Hall, 1999.

[Rah01] E. Rahm, P. Bernstein, "On Matching Schemas Automatically", *Tech. Report 1/2001,* Comp. Science Dept., U. Leipzig, Feb. 2001 http://dol.uni-leipzig.de/pub/2001-5, to appear in *VLDB Journal.*

[Ros97] A. Rosenthal, E. Sciore, S. Renner, "Toward Integrated Metadata for the Department of Defense", *IEEE Metadata Workshop,* Silver Spring, MD, 1997. http://www.mitre.org/pubs/data_mgt/Papers/arosenthal.pdf

[Ros00]A. Rosenthal, F. Manola, S. Renner, "Getting Data to Applications: Why We Fail, and How We Can Do Better", *AFCEA Federal Database Conf.*, Sept. 2000.

[Sel01] L. Seligman, A. Rosenthal, "The Impact of XML on Databases and Data Sharing", IEEE Computer, June 2001, pp. 59-67.

[Yan01] L. Yan, R. Miller, L. Haas, R. Fagin, "Data-Driven Understanding and Refinement of Schema Mappings". *ACM SIGMOD Conf.*, Santa Barbara, 2001, pp. 485-496.

## Author Biographies

**Dr. Arnon Rosenthal** (http://www.mitre.org/resources/centers/it/staffpages/arnie/) is a principal scientist at The MITRE Corp, and has a PhD from Berkeley. His research interests include data administration, distributed object management, legacy system migration, and data security. He served on the program committee for the 2001 Very Large Data Base and ACM SIGMOD conferences. He has been on the faculty at U. Michigan, and a senior computer scientist at Computer Corporation of America.

**Dr. Len Seligman** is a principal scientist at The MITRE Corp. and an associate editor of *SIGMOD Record,* the quarterly bulletin of the ACM Special Interest Group on the Management of Data. He is also the Industry Track Chair for the 2001 ACM International Conference on Information and Knowledge Management (CIKM). His research interests include data interoperability, semistructured data, and integration of artificial intelligence and database technologies. Seligman received a PhD in Information Technology from George Mason University.