View-based Data Integration

Yannis Katsis
Yannis Papakonstantinou
Computer Science and Engineering
UC San Diego, USA
{ikatsis,yannis}@cs.ucsd.edu

DEFINITION

Data Integration (or *Information Integration*) is the problem of finding and combining data from different sources. *View-based Data Integration* is a framework that solves the data integration problem for *structured* data by integrating sources into a single unified view. This integration is facilitated by a declarative *mapping language* that allows the specification of how each source relates to the unified view. Depending on the type of view specification language used, view-based data integration systems (VDISs) are said to follow the *Global As View (GAV)*, *Local As View (LAV)* or *Global and Local As View (GLAV)* approach.

HISTORICAL BACKGROUND

Data needed by an application are often provided by a multitude of data sources. The sources often employ heterogeneous data formats (e.g. text files, web pages, XML documents, relational databases), structure the data in different ways and can be accessed through different methods (e.g. web forms, database clients). This makes the task of combining information from multiple sources particularly challenging. To carry it out, one has to retrieve data from each source individually, understand how the data of the sources relate to each other and merge them, while accounting for discrepancies in the structure and the values, as well as for potential inconsistencies.

The first to realize this problem were companies willing to integrate their structured data within or across organizations. Soon the idea of integrating the data into a single unified view emerged. These systems, to be referred as view-based integration systems (VDISs) would provide a single point of access to all underlying data sources. Users of a VDIS(or applications) would query the unified view and get back integrated results from all sources, whereas the task of combining data from the sources and resolving inconsistencies would handled by the system transparently to the applications.

The VDISs made their first appearance in the form of multidatabases and federated systems [11]. Subsequently the research community dive into the problem of specifying the correspondence between the sources and the unified view. The outcome were three categories of languages to express the correspondence (GAV, LAV and GLAV) together with several related theoretical and system results. The industry also embraced the view-based integration framework by creating many successful VDISs (e.g. BEA AquaLogic, IBM WebSphere).

SCIENTIFIC FUNDAMENTALS

Abstracting out the differences between individual systems, a typical VDIS conforms to the architecture shown in Figure 1. Sources store the data in a variety of formats (relational databases, text files etc.). Wrappers solve the heterogeneity in the formats by transforming each source's data model to a common data model used by the integration system. The wrapped data sources are usually referred to as local or source databases, the structure of which is described by corresponding local/source schemas. This is in contrast to the unified view exported by the mediator, also called global/target database. Finally mappings expressed in a certain mapping language (depicted as lines between the wrapped sources and the mediator) specify the relationship between the wrapped data sources (i.e. the local schemas) and the unified view exported by the

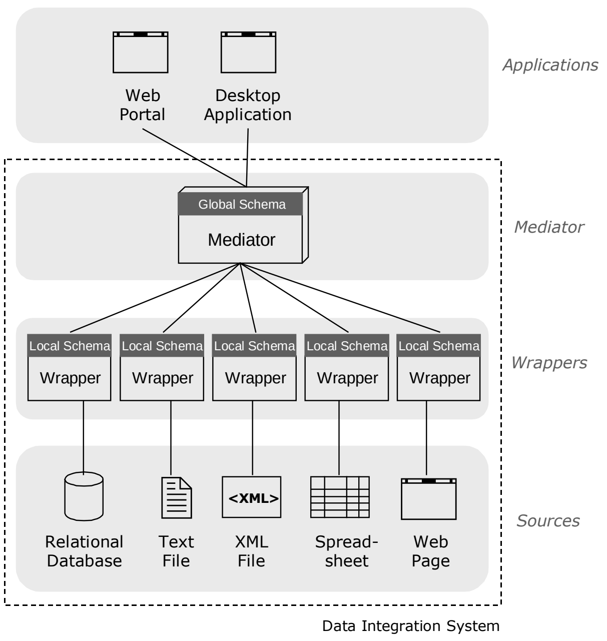

Figure 1: View-based Data Integration System (VDIS) Architecture

mediator (i.e. the global schema).
Given a VDIS, applications or users retrieve data from the sources indirectly by querying the global schema. It is the task of the mediator to consult the mappings to decide which data to retrieve from the sources and how to combine them appropriately in order to form the answer to the user's query.

VDISs can be categorized according to the following three main axes:

Common data model and query language. The data model and query language that is exposed by the wrappers to the mediator and by the mediator to the applications. Commonly used data models include the relational, XML and object-oriented data model.

- *Mapping Language.* The language for specifying the relationship of sources with the view. Languages proposed in the literature fall into three categories; *Global As View (GAV)*, *Local As View (LAV)* and *Global and Local As View (GLAV)*. Being one of the most important components in a VDIS, these are discussed in detail below.
- *Data storage method.* The decision on the place where the data are actually stored. The two extremes are the materialized and the virtual approach (see [14] for a comparison). In the materialized integration (also known as eager, in-advance or warehousing approach), all source data are replicated on the mediator. On the other hand, in the virtual mediation (e.g. Infomaster [6]) (or lazy approach), the data are kept in the sources and the global database is virtual. Consequently, a query against the global database cannot be answered directly but has to be translated to queries against the actual sources. Finally, some systems employ hybrid policies, such as virtualization accompanied by a cache .

1 Specifying the Relationship of the Sources with the Unified View

To allow the mediator to decide which data to retrieve from each source and how to combine them into the unified view, the administrator of the VDIShas to specify the correspondence between the local schema of each source and the global schema through mappings.
The mappings are expressed in a language, corresponding to some class of logic formulas. Languages proposed in the literature fall into three categories: *Global As View (GAV)*, *Local As View (LAV)* and *Global and Local As View (GLAV)*. In GAV the global database (schema) is expressed as a function of the local databases (schemas). LAV on the other hand follows the opposite direction with each local schema being described as a function over the global schema. Therefore LAV allows to add a source to the system independently of other sources. Finally GLAV is a generalization of the two.

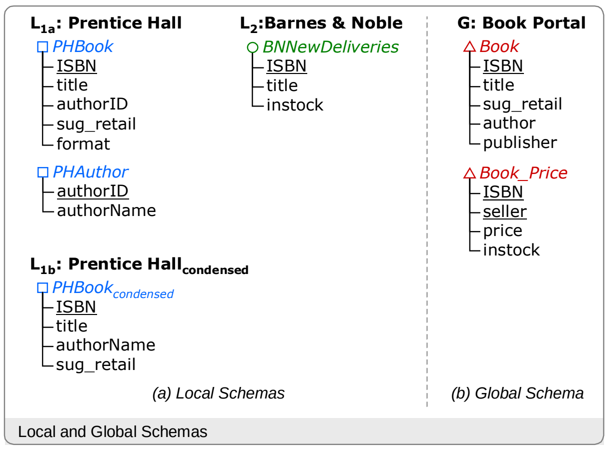

Figure 2: Local and Global Schemas of the running example

This section presents each of these approaches in detail and explains their implications on the query answering algorithms. Essentially each of them represents a different trade-off between expressivity and hardness in query answering.

Running Example. *For ease of exposition, the following discussion employs a running example of integrating information about books. The example employs the relational data model for both the sources and the global database. Moreover the query language used by the users to extract information from the global database and in turn by the mediator to retrieve information from the sources is the language of conjunctive queries with equalities ($CQ^=$); a subset of SQL widely adopted in database research.*
Figure 2 shows the employed local and global schemas. Relations are depicted in italics and their attributes appear nested in them. For instance, the global schema $\mathcal{G}$ in Figure 2b consists of two relations Book and Book_Price. Relation Book has attributes ISBN, title, suggested retail price, author and publisher, while Book_Price stores the book price and stock information for different sellers. Data is fueled into the system by two sources; the databases of the bookstore Barnes & Noble (B&N) and the publisher Prentice Hall (PH), with the schemas shown in Figure 2a. Note that there are two versions of the PH schema, used in different examples.

1.1 Global As View (GAV)

Historically the first VDISs followed the Global As View (GAV) approach ([7, 12]), in which the global schema is described in terms of the local schemas. In such systems the contents of each relation R in the target schema $\mathcal{G}$ are specified through a query (view) V over the combined schemas of the sources.
Thus in GAV the correspondence between the local schemas and the global schema can be described through a set of mappings of the form

$$V_i \rightarrow I(R_i)$$

one for each relation R_i of the global schema. $I(R_i)$ is the identity query over R_i (i.e. a query that returns all attributes of R_i)

Example: Consider the following two GAV mappings [1]*:*
$M_1 : V_1 \rightarrow I(Book).$
$M_2 : V_2 \rightarrow I(Book\_Price)$

where $V_1(ISBN, title, sug\_retail, authorName, \text{“}PH\text{”}) :- PHBook(ISBN, title, authorID, sug\_retail, format),$
$PHAuthor(authorID, authorName)$
and $V_2(ISBN, \text{“}B\&N\text{”}, sug\_retail, instock) :- PHBook(ISBN, title, authorID, sug\_retail, format),$

[1]For the examples, the identity query I over some relation is considered to return the attributes of that relation in the same order as they appear on the schema in Figure 2

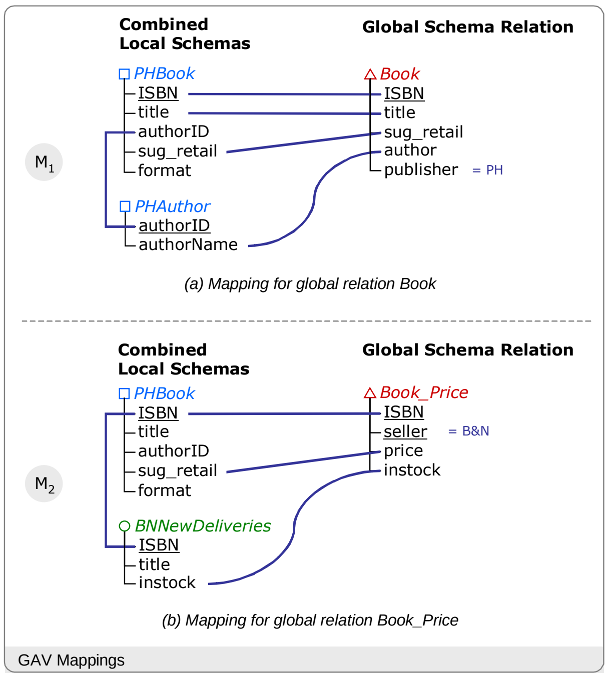

Figure 3: Example of GAV Mappings

$$BNNewDeliveries(ISBN, title, instock)$$

The mappings are graphically depicted in Figure 3a and b, as described in [9]. This is similar to the way most mapping tools *(i.e. tools that allow a visual specification of mappings), such as IBM Clio and MS BizTalk Mapper, display mappings. Mapping M_1 intuitively describes how Book tuples in the global database are created. This is done by retrieving the ISBN, title and sug_retail from a PHBook tuple, the author from the corresponding PHAuthor tuple (i.e. the PHAuthor tuple with the same authorID as the PHBook tuple) and finally setting the publisher to "PH"(since the extracted books are published by PH).*
Similarly, mapping M_2 describes the construction of the global relation Book_Price. This involves combining information from multiple sources: the price from the suggested retail price information provided by PH and the inventory information from B&N, because B&N's administrator knows that B&N's sells its books at the suggested retail price.

Query Answering in GAV. GAV mappings have a *procedural* flavor, since they describe how the global database can be constructed from the local databases. For this reason, query answering in GAV is straightforward, both in the materialized and the virtual approach.
In the materialized approach, the source data are replicated in the global database by executing for each mapping $V_i \rightarrow I(R_i)$ the query V_i against the local databases and populating R_i with the query results. Subsequently an application query Q against the global schema is answered by simply running Q over the materialized global database.
On the other hand, in the virtual approach data are kept in the sources and thus a query against the global schema has to be translated to corresponding queries against the local schemas. Due to the procedural flavor of GAV, this can be done through view unfolding (i.e. replacing each relational atom of the global schema in the query by the corresponding view definition). Intuitively whenever a query asks for a global relation R_i, it will instead run the subquery V_i over the local schemas, which, according to the mapping $V_i \rightarrow I(R_i)$ provides the contents of R_i.

Advantages. The simplicity of the GAV rules together with the straight-forward implementation of query answering, led to the wide adoption of GAV by industrial systems. From the research sector representative GAV-based VDISs systems are MULTIBASE [11], TSIMMIS [5] and Garlic [2].

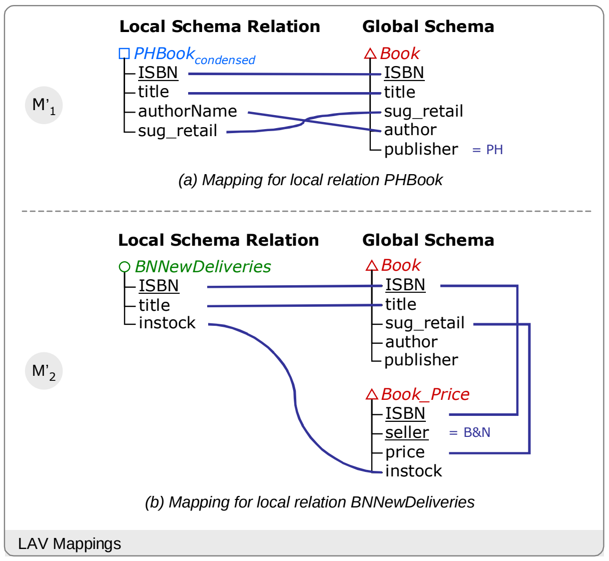

Figure 4: Example of LAV Mappings

Disadvantages. However GAV has also several drawbacks:
First, since the global schema is expressed in terms of the sources, *global relations cannot model any information not present in at least one source.* For instance, the Book relation in the example could not contain an attribute for the book weight, since no source currently provides it. In other words, the value of each global attribute has to be explicitly specified (i.e. in the visual representation all global attributes must have an incoming line or an equality with a constant).
Second, as observed in mapping M_2 of the running example, *a mapping has to explicitly specify how data from multiple sources are combined to form global relation tuples.* Therefore GAV-based systems do not facilitate adding a source to the system independently of other sources. Instead when a new source wants to join the system, the system administrator has to inspect how its data can be merged with those of the other sources currently in the system and modify the corresponding mappings.

1.2 Local As View (LAV)

To overcome the shortcomings of GAV, researchers came up with the Local As View (LAV) approach ([7, 12]). While in GAV the global schema is described in terms of the local schemas, LAV follows the opposite direction expressing each local schema as a function of the global schema. LAV essentially corresponds to the "source owners view" of the system by describing which data of the global database are present in the source. Using the same notation as in GAV, local-to-global correspondences can be written in LAV as a set of mappings:

$$I(R_i) \rightarrow U_i$$

one for every relation R_i in the local schemas, where U_i is a query over the global schema and I the identity query.

Example: Figure 4 shows the following two LAV mappings for the running example:
$M_1' : I(PHBook_{condensed}) \rightarrow U_1$
$M_2' : I(BNNewDeliveries) \rightarrow U_2$

where $U_1(ISBN, title, author, sug\_retail) :-\quad Book(ISBN, title, sug\_retail, author, \text{“}PH\text{”})$
and $\quad U_2(ISBN, title, instock) :-\quad Book(ISBN, title, sug\_retail, author, publisher),$
$\qquad Book\_Price(ISBN, \text{“}B\&N\text{”}, sug\_retail, instock)$

For instance, M_1' *specifies that* $PHBook_{condensed}$ *contains information about books published by PH. Similarly,* M_2' *declares that* $BNNewDeliveries$ *contains the ISBN, title of books sold by B&N at their suggested retail price and* *whether B&N has them in stock.*

In contrast to GAV mappings, LAV mappings have a *declarative* flavor, since, instead of explaining how the global database can be created, they describe what information of the global database is contained in each local database.

Advantages. LAV addresses many of GAV problems with the most important being that sources can register independently of each other, since a source's mappings do not refer to other sources in the system.

Disadvantages. However, LAV suffers from the symmetric drawbacks of GAV. In particular it cannot model sources that have information not present in the global schema (this is the reason why the example above used the condensed version of PH's schema that did not contain the attribute format, which is not present in the global schema). Furthermore, due to LAV's declarative nature, query answering is non-trivial any more, as described next. Mainly because of its technical implications to query answering, LAV has been extensively studied in the literature. Representative LAV-based systems include Information Manifold [10] and the system described in [13].

Query Answering in LAV. Since LAV mappings consist of an arbitrary query over the global schema, they may leave some information of the global database unspecified. For instance, mapping M_2' above only states that B&N sells its books at the suggested retail price, without specifying the exact price. Thus there might be infinitely many global databases that could be inferred from the sources through the mappings (each of them would make sure that each pair of Book and Book_Price tuples created from a BNNewDeliveries tuple share the same value for price but each such global database might choose a different constant for the value). These databases are called *possible worlds*. Their existence has two important implications: First, since there does not exist a unique global database, it cannot be materialized and therefore LAV lends itself better to virtual mediation. However, there is still a way of replicating source information in a centralized place. This involves creating a "special" database that intuitively stores the general shape of all possible worlds. This "special" database is called *canonical universal solution* and can be built through procedures employed in data exchange ([3]).
Second, since there exist many global databases, the query answering semantics need to be redefined. The standard semantics adopted in the literature for query answering in LAV-based systems are based on the notion of *certain answers* ([1, 8]). The certain answers to a query are the answers to the query, which will always appear regardless of which possible world the query is executed against (i.e. the tuples that appear in the intersection of the sets of query answers against each possible world). Intuitively certain answers return information that *is guaranteed* to exist in *any* possible world.

Example: For example, if in the integration system of Figure 4 a query asks for all ISBNs that are sold by some seller at their suggested retail price, it will get back all ISBNs stored in relation BNNewDeliveries, because for each of them, any possible world will contain a pair of Book and Book_Price tuples with the same ISBN and the same prices (although these prices will have different values between possible worlds). On the other hand, if the query asks for all books sold at a specific price, it will not get back the ISBNs from BNNewDeliveries, because their exact prices are left unspecified by the mapping M_2' and will therefore differ among possible worlds.

In order to compute the certain answers to a query in a virtual integration system following the LAV approach, the query against the global schema has to be translated to corresponding queries against the local schemas. This problem is called *rewriting queries using views* (because the query over the global database has to be answered by using the sources which are expressed as views over it) and is of interest also to other areas of database research, such as query optimization and physical data independence (see [8] for a survey). In contrast to GAV, it is a non-trivial problem studied extensively by researchers.

1.3 Global and Local As View (GLAV)

To overcome the limitations of both GAV and LAV, [4] proposed a new category of mapping languages, called Global and Local As View (GLAV), which is a generalization of both GAV and LAV. GLAV mappings are of the form:

$$V_i \rightarrow U_i$$

where V_i, U_i are queries over the local and global schema, respectively.

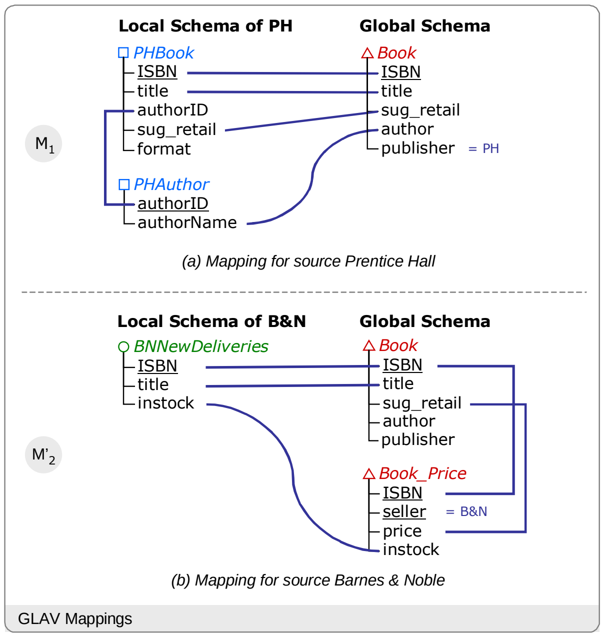

Figure 5: Example of GLAV Mappings

GLAV languages can trivially express both GAV mappings and LAV mappings by assigning to U_i a query returning a single global relation or to V_i a query asking for a single local relation, respectively. However GLAV is a strict superset of both GAV and LAV by allowing the formulation of mappings that do not fall either under GAV or under LAV (i.e. mappings in which V_i and U_i both do not return just a single local or global relation).

Example: Figure 5 shows two GLAV mappings. The first mapping is the GAV mapping M_1 first presented in Figure 3a, while the second mapping is the LAV mapping M_2' used in Figure 4b.

Since U_i (a.k.a. the conclusion of the mapping) can be an arbitrary query over the global schema, GLAV allows the independent registration of sources in the same way as LAV. However this also implies that the global database is incomplete. Consequently, query answering in GLAV is usually done under certain answer semantics, by extending query rewriting algorithms for LAV ([15]).

2 Alternatives to VDISs

Apart from the VDISs, research and industrial work led to many alternative approaches to data integration of structured data:

Vertical Integration Systems are specialized applications that solve the problem of data integration for a specific domain. For example, www.mySimon.com or www.rottentomatoes.com integrate price and movie information, respectively.

•*Extract Transform Load (ETL) Tools* generally facilitate the actual migration of data from one system to another. When used for data integration they are closely tied to the problem of materializing the integrated database in a central data warehouse.

Compared to these solutions, VDISs offer a more general approach to data integration with the following advantages: (a) The relationships between the sources and the unified view are explicitly stated and not hidden inside a particular implementation, (b) a general VDIS implementation can be used in many different domains and (c) a VDIS deployment can be easily utilized by many applications, as shown in Figure 1. In a way VDISs are analogous to Database Management Systems, offering a general way of managing the data (which in VDISs are heterogeneous and distributed), independently of the applications that need those data.

Finally, another alternative to VDISa are Peer-to-Peer (P2P) Integration Systems that drop the requirement for a single unified view, allowing queries to be posed over any source schema. Although it is an active area of research, P2P systems have not yet been widely adopted in industry.

KEY APPLICATIONS

VDISs are used for integration of structured data in many different settings, including among others enterprises, government agencies and scientific communities.

CROSS REFERENCE

Information Integration, Peer-to-Peer Data Integration, Query Rewriting Using Views, Schema Mapping

RECOMMENDED READING

Between 3 and 15 citations to important literature, e.g., in journals, conference proceedings, and websites.

[1] Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *PODS*, pages 254–263, 1998.

[2] Michael J. Carey, Laura M. Haas, Peter M. Schwarz, Manish Arya, William F. Cody, Ronald Fagin, Myron Flickner, Allen Luniewski, Wayne Niblack, Dragutin Petkovic, Joachim Thomas II, John H. Williams, and Edward L. Wimmers. Towards heterogeneous multimedia information systems: The Garlic approach. In *RIDE-DOM*, pages 124–131, 1995.

[3] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *ICDT '03: Proceedings of the 9th International Conference on Database Theory*, pages 207–224, London, UK, 2002. Springer-Verlag.

[4] Marc Friedman, Alon Levy, and Todd Millstein. Navigational plans for data integration. In *AAAI*, 1999.

[5] H.K. Garcia-Molina, Y.K. Papakonstantinou, D.K. Quass, A.K. Rajaraman, Y.K. Sagiv, J.K. Ullman, V.K. Vassalos, and J.K. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

[6] Michael R. Genesereth, Arthur M. Keller, and Oliver M. Duschka. Infomaster: An information integration system. In *SIGMOD*, 1997.

[7] Alon Halevy. Logic-based techniques in data integration. In *Logic Based Artificial Intelligence*, 2000.

[8] Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.

[9] Yannis Katsis, Alin Deutsch and Yannis Papakonstantinou. Interactive Source Registration in Community-oriented Information Integration. In *VLDB* 2008.

[10] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *Information Gathering from Heterogeneous, Distributed Environments*, 1995.

[11] Terry Landers and Ronni L. Rosenberg. An overview of MULTIBASE. *Distributed systems, Vol. II: distributed data base systems table of contents*, pages 391–421, 1986.

[12] Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, 2002.

[13] I. Manolescu, D. Florescu, and D. Kossmann. Answering XML queries over heterogeneous data sources. In *VLDB*, 2001.

[14] Jennifer Widom. Research problems in data warehousing. In *CIKM*, 1995.

[15] Cong Yu and Lucian Popa. Constraint-based XML query rewriting for data integration. In *SIGMOD*, 2004.