Using Schema Mapping to Facilitate Data Integration

Li Xu* and David W. Embley*
Department of Computer Science
Brigham Young University
Provo, Utah 84602, U.S.A.
{lx, embley}@cs.byu.edu

Abstract

To integrate data from disparate, heterogeneous information sources in an open environment, data-integration systems demand a resolution of several major issues: (1) heterogeneity, (2) scalability, (3) continual infusion and change of local information sources, (4) query processing complexity, and (5) global schema evolution. Although several data-integration systems have been proposed to address these problems, no single system addresses all the issues in a unified approach. To resolve these problems, we offer TIQS, an approach to data integration that uses semi-automatic schema matching to produce source-to-target mappings based on a predefined conceptual target schema. In a unified approach, TIQS offers solutions for each of the five major issues. Compared with other data integration approaches, our approach combines their advantages, mitigates their disadvantages, and provides a viable alternative for flexible and scalable data integration.

1 Introduction

Data integration refers the problem of combining data residing at autonomous and heterogeneous sources and providing users with a unified global schema [Ull97, Hal01, CCGL02]. Two main concepts constitute the architecture of a data-integration system [Ull97]: wrappers and mediators. A *wrapper* wraps an information source and models the source using a *source schema*. A *mediator* maintains a *global schema* and *mappings* between the global and source schemas. We focus here on data-integration systems that do not materialize data in the global schema. Currently, there are two main initiatives to integrate data and answer queries without materializing a global schema: Global-as-view (GAV) [CGMH+94] and Local-as-View (LAV) [LRO96, GKD97]. In either a GAV or LAV approach, whenever a user poses a query in terms of relations in the global schema, the

*This material is based upon work supported by the National Science Foundation under grant IIS-0083127.

mediator within the data-integration system uses a *query-reformulation* procedure to translate the query into sub-queries that can be executed in sources such that the mediator can collect returned answers from the sources and combine them as the answer to the query.

One of the important applications for data integration is to deal with the explosion of data on the World Wide Web. E-business applications such as comparison shopping and knowledge-gathering applications such as vacation planning raise the following major issues for approaches to data integration. (1) *Heterogeneity.* The sources are autonomous, establishing their own vocabulary and structure. (2) *Scalability.* The number of sources to access and integrate is large. (3) *Continual infusion and change of local information sources.* New sources continually become available and become part of the system. Sources within the system may change frequently. (4) *Query processing complexity.* Users frequently pose queries, which can be complex with respect to the collection of information sources. (5) *Evolution.* As applications evolve, Database Administrators (DBAs) may wish to change the global schema to include some new items of interest. Most of the existing approaches to data integration [CGMH+94, LRO96, GKD97, Ull97, FLM99, MHH+01, CCGL02], however, have addressed only some of these issues. Thus, they do not meet the needs of the modern E-business applications.[1] To address these issues, we present an alternative point of view, called TIQS (Target-based Integration Query System).

The following five characteristics describe TIQS. Each characteristic addresses one of the five major issues.

1. *Heterogeneity.* Each relation in a target schema, which is our global schema, is predefined and independent of any source schema. Moreover, we wrap sources in isolation, without reference to the global schema.[2] Thus, in TIQS, source and target schemas use different

[1] After describing our proposed solution we explain in Section 6, by way of comparison, how other proposed solutions fail to address one or more of these issues.

[2] Often these sources are structured, and we simply take the local schema without change [ETL02].

structures and vocabularies. A mapping tool [XE03] within TIQS produces a set of mapping elements semi-automatically in a source-to-target mapping that maps a source schema to a target schema. The mapping elements include both direct and many indirect semantic correspondences. Thus, TIQS reduces heterogeneity by applying semantic correspondences expressed using source-to-target mappings between target and source schemas.

2. *Scalability.* Although TIQS still requires a DBA to validate and sometimes adjust the generated source-to-target mappings, TIQS largely automates this mapping procedure [XE03]. This facilitates scalability—allows TIQS to automatically specify views over a large number of source schemas that match with elements in the target schema.

3. *Continual infusion and change of local information sources.* When a new information source becomes available (changes), a source-to-target mapping must be created (adjusted). With the assistance of the semi-automatic mapping tool in TIQS, the maintenance requires little manual work to create (adjust) mappings.

4. *Query processing complexity.* Whenever a user poses queries in terms of target relations, TIQS uses its generated source-to-target mappings to reduce query reformulation to simple rule unfolding (standard execution of views in ordinary databases). This reduces query processing complexity.

5. *Evolution.* If the target schema evolves, the mapping tool of TIQS semi-automatically generates (or adjusts) mapping elements between the new target schema and the source schemas.

TIQS operates in two phases: design and query processing. In the design phase, the system synergistically automates the generation of source-to-target mappings. Mapping elements in source-to-target mappings are expressions over source *schema elements* that produce *virtual target-view elements*. This leads automatically to a rewriting of every target element as a union of corresponding

virtual target-view elements. In the query processing phase, a user poses queries in terms of target relations. Query reformulation thus reduces to rule unfolding by applying the view definition expressions for the target relations in the same way database systems apply view definitions.

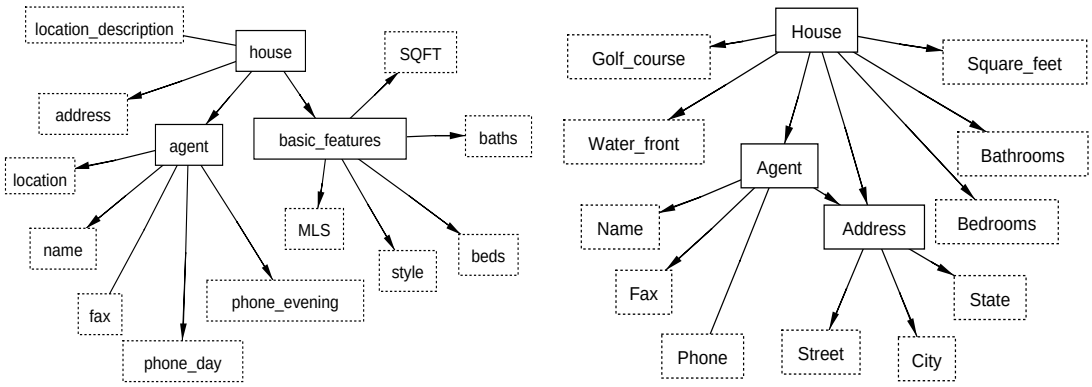
The contributions of TIQS are: (1) a unified, scalable approach to data integration using source-to-target mappings based on a predefined target schema in which query reformulation reduces to query unfolding, (2) a practical approach whose implementation is readily available based on schema-matching techniques described in [XE03], and (3) a correlation of research work on schema matching [RB01] and data integration [Ull97], which are usually considered to be orthogonal. Beyond previous work [EJX01, BE03, XE03], this paper unifies all the components of TIQS, presents a formal representation of source-to-target mappings, reduces query-reformulation complexity based on source-to-target mappings, and proves a theorem for *sound* and *maximal* query answers to user queries.

We organize the contributions in this paper as follows. Section 2 presents the components of TIQS. Section 3 describes the matching techniques to generate and the expression language to represent source-to-target mappings. Section 4 shows an example of how to add a new information source into TIQS. Section 5 discusses the solution to query reformulation and gives a theorem to prove that TIQS returns all available answers to a query. Section 6 reviews and compares alternatives to TIQS. In Section 7 we summarize and make concluding remarks.

2 The Data Integration System

Definition 1. A *data-integration system* I is a triple $(T, \{S_i\}, \{M_i\})$, where T is a target schema, $\{S_i\}$ is a set of n source schemas, and $\{M_i\}$ is a set of n source-to-target mappings, such that for each source schema S_i there is a mapping M_i from S_i to T, $1 \leq i \leq n$.

We use rooted graphs to represent both target and source schemas in I. A graph includes a



(a) Schema 1 (b) Schema 2

Figure 1: Source Graphs for Schema 1 and Schema 2

set of *object sets* O and a set of *relationship sets* R. Therefore, a *schema element* is either an object set or a relationship set. An object set either has associated data values or has associated object identifiers (OIDs), which we respectively call *lexical* and *non-lexical* object sets. The root node is a designated non-lexical object set of primary interest. Figure 1, for example, shows two schema graphs (whose roots are *house* and $House$). In the graphs, lexical object sets are dotted boxes, non-lexical object sets are solid boxes, functional relationship sets are lines with an arrow from domain object set to range object set, and nonfunctional relationship sets are lines without arrowheads. For a schema H, which is either a source schema or a target schema, we let Σ_H denote the union of O and R. For source views, we let V_H denote the extension of Σ_H with *derived object and relationship sets*, each of which corresponds a view defined over source H. We call derived object and relationship sets in V_H *virtual elements* of source H.

A source-to-target mapping M_i for a source schema S_i with respect to a target schema T is a function $f_i(V_{S_i}) \rightarrow \Sigma_T$. Intuitively, a source-to-target mapping M_i represents inter-schema correspondences between a source schema S_i and a target schema T. If we let Schema 1 in Figure 1(a) be the target and let Schema 2 in Figure 1(b) be the source, for example, a source-to-target mapping between the two schemas includes a semantic correspondence, which declares that the lexical object

set *Bedrooms* in the source semantically corresponds to the lexical object set *beds* in the target. If we let Schema 1 be the source and Schema 2 be the target, a source-to-target mapping declares that the union of the two sets of values in *phone_day* and *phone_evening* in the source corresponds to the values for *Phone* in the target.

We represent semantic correspondences between a source schema S and a target schema T as a set of *mapping elements*. A mapping element is either a *direct match* which binds a schema element in Σ_S to a schema element in Σ_T, or an *indirect match* which binds a derived schema element in V_S to a target schema element in Σ_T through an appropriate *mapping expression* over Σ_S. A mapping expression specifies how to derive a virtual element through manipulation operations over a source schema. For either a direct match or an indirect match, the source schema element is a virtual target-view element for the target schema element in the match. We denote a mapping element as $(t \sim s \Leftarrow \theta_s(\Sigma_S))$, where $\theta_s(\Sigma_S)$ is a mapping expression that derives a virtual target-view element s in V_S, and t is a target schema element in Σ_T. Note that the mapping expression may be degenerate so that $(t \sim s)$ is possible.

As part of the mapping declarations, TIQS derives a set of *inclusion dependencies* for each target element based on the collected source-to-target mappings. Each mapping element ω, $(t \sim s \Leftarrow \theta_s(\Sigma_S))$, implies an inclusion dependency, which we denote as $(S.s \subseteq t)$. This declares that the facts for schema element $s \in V_S$, can be "loaded" into the target as the facts for schema element t.[3] As is typical for integration systems with non-materialized global schemas, we make an "Open World Assumption." Thus, the facts for the source element s in the mapping element ω are only a subset of facts for the target element t; and if there exists a source element $s' \in V_{S'}$ and another mapping element ω', $(t \sim s' \Leftarrow \theta_{s'}(\Sigma_{S'}))$, the facts for both s and s' can be facts for t. In general, for each target schema element $t \in \Sigma_T$ in the data-integration system I, we denote the

[3]For TIQS and other integration systems, whose global schema is virtual, the "loading" is implicit.

set of inclusion dependencies for t as $\{S_i.s_j \subseteq t | (t \sim s_j \Leftarrow \theta_{s_j}(\Sigma_{S_i})) \in M_i, s_j \in V_{S_i}, S_i \in \{S_i\} \in I, M_i \in \{M_i\} \in I, T \in I\}$.

3 Source-to-Target Mappings

Automated *schema matching* techniques have been proven to be successful in extracting mapping elements between two schemas. [RB01] surveys these techniques. The mapping tool in TIQS provides many mappings automatically, with accuracies ranging from 92%-100%; these mappings are not just direct matches, but include many indirect matches discussed later in this paper.

3.1 Matching Techniques

The mapping tool uses four basic techniques for matching: (1) terminological relationships (e.g. synonyms and hypernyms), (2) data-value characteristics (e.g. string lengths and alphanumeric ratios), (3) domain-specific, regular-expression matches (i.e. the appearance of expected strings), and (4) structure (e.g. structural similarities). In the following we give high-level descriptions of the four matching techniques we use in the mapping tool. (See details in [XE03].)

- *Terminological Relationships.* The meaning of element names provides a clue about which elements match. To match element names, we use WordNet [Mil95] which organizes English words into synonym and hypernym sets. Other researchers have also suggested using WordNet to match attributes (e.g. [BCV99]), but have given few, if any, details. We use a C4.5 [Qui93] learning algorithm to train a set of decision rules to compute confidence values each of which gives the possibility that a source schema element matches with a target schema element.

- *Data-Value Characteristics.* Whether two sets of data have similar value characteristics provides another a clue about which elements match. Previous work in [LC00] shows that this technique can successfully help match elements by considering such characteristics as string-lengths and alphabetic/non-alphabetic ratios of alphanumeric data and means and variances

of numerical data. We use features similar to those in [LC00], but generate a C4.5 decision tree rather than a neural-net decision rule. Based on the decision tree, we generate confidence values between two schema elements with respect to their value characteristics.

- *Expected Data Values.* Whether expected values appear in a set of data provides yet another clue about which elements match. For a specific application, we can specify a domain ontology [ECJ+99], which includes a set of concepts and relationships among the concepts and a set of regular expressions that match values and keywords expected to appear for the concepts. Then, using techniques described in [ECJ+99], we can extract values from sets of data associated with source elements and categorize their data-value patterns based on the regular expressions declared for target-application concepts. The derived data-value patterns and the declared relationship sets among concepts in the domain ontology can help discover both direct and indirect matches for schema elements.

- *Structure.* We consider structure matching as one more technique that provides a clue about which elements match. Given the confidence measures output from the other matching techniques as a guide, structure matching determines element matches by considering the context of schema elements and exploiting structure properties of target and source schemas. The structure-matching technique is the other key technique in addition to *Expected Data Values* that can help us discover indirect as well as direct matches for schema elements.

3.2 Extended Relational Algebra

Each object and relationship set (including derived object and relationship sets) in the target and source schemas are single-attribute or multiple-attribute relations. Thus, relational algebra directly applies to the object and relationship sets in a source or target schema. The standard operations, however, are not enough to capture the operations required to express all the needed

source-to-target mappings. Thus, we extend the relational algebra.

To motivate our use of standard and extended operators, we list the following problems we must face in creating derived object and relationship sets over source schemas.

- *Union* and *Selection*. The object sets, *phone_day* and *phone_evening* in Schema 1 of Figure 1(a) are both subsets of *Phone* values in Schema 2 of Figure 1(b), and the relationship sets $agent - phone\_day$ and $agent - phone\_evening$ in Schema 1 are both specializations of $Agent - Phone$ value pairs in Schema 2. Thus, if Schema 2 is the target, we need the union of the values in *phone_day* and *phone_evening* and the union of the relationships in $agent - phone\_day$ and $agent - phone\_evening$ in Schema 1; and if Schema 1 is the target, we should use *Selection* to find a way to separate the day phones from the evening phones and separate the relationships between agents and day phones from those between agents and evening phones.

- *Merged* and *Split Values*. The object sets, *Street*, *City*, and *State* are separate in Schema 2 and merged as *address* of *house* or *location* of *agent* in Schema 1. Thus, we need to split the values if Schema 2 is the target and merge the values if Schema 1 is the target.

- *Object-Set Name as Value*. In Schema 2 the features $Water\_front$ and $Golf\_course$ are object-set names rather than values. The Boolean values "Yes" and "No" associated with them are not the values but indicate whether the values $Water\_front$ and $Golf\_course$ should be included as description values for *location_description* of *house* in Schema 1. Thus, we need to distribute the object-set names as values for *location_description* if Schema 1 is the target and make Boolean values for $Water\_front$ and $Golf\_course$ based on the values for *location_description* if Schema 2 is the target.

- *Path as Relationship Set*. The path $house - basic\_features - beds$ in Schema 1 semantically

corresponds to the relationship set $House-Bedrooms$ in Schema 2. Thus, we need to join and project on the path if Schema 2 is the target and make a derived object set for $basic\_features$ and derived relationship sets for $house - basic\_features$ and $basic\_features - beds$ over Schema 2 if Schema 1 is the target.

Currently, we use the following operations over source relations to resolve these problems[4]. (See Appendix A for examples that illustrate how the new operators work.)

- *Standard Operators. Selection* σ, *Union* $\cup$, *Natural Join* $\bowtie$, *Projection* π, and *Rename* ρ.

- *Composition* λ. The λ operator has the form $\lambda_{(A_1,\ldots,A_n),A}r$ where each A_i, $1 \leq i \leq n$, is either an attribute of r or a string, and A is a new attribute. Applying this operation forms a new relation r', where $attr(r') = attr(r) \cup \{A\}$ and $|r'| = |r|$. The value of A for tuple t of row l in r' is the concatenation, in the order specified, of the strings among the A_i's and the string values for attributes among the A_i's for tuple t' of row l in r.

- *Decomposition* γ. The γ operator has the form $\gamma^R_{A,A'}r$ where A is an attribute of r, and A' is a new attribute whose values are obtained from A values by applying a routine R. Typically, R extracts a substring from a given string to form part of a decomposition. Repeated application of γ allows us to completely decompose a string. Applying this operation forms a new relation r', where $attr(r') = attr(r) \cup \{A'\}$ and $|r'| = |r|$. The value of A' for tuple t of row l in r' is obtained by applying the routine R on the value of A for tuple t' of row l in r.

- *Boolean* β. The β operator has the form $\beta^{Y,N}_{A,A'}r$, where Y and N are two constants representing Yes and No values in r, A is an attribute of r that has only Y or N values, and A' is a new attribute. The β operator requires the precondition $(attr(r) - \{A\}) \rightarrow \{A\}$. Applying this

[4]In the notation, a relation r has a set of attributes, which corresponds to the names of lexical or non-lexical object sets; $attr(r)$ denotes the set of attributes in r; and $|r|$ denotes the number of tuples in r.

operation forms a new relation r', where $attr(r') = (attr(r) - \{A\}) \cup \{A'\}$ and $|r'| = |\sigma_{A=Y} r|$. The value of A' for tuple t in r' is the literal string A if and only if there exists a tuple t' in r such that $t'[attr(r) - \{A\}] = t[attr(r) - \{A\}]$ and $t'[A]$ is a Y value.

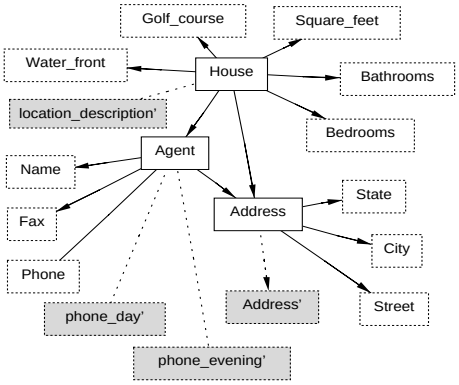
- *DeBoolean* $\eth$. The $\eth$ operator has the form $\eth^{Y,N}_{A,A'} r$, where Y and N are two constants representing Yes and No values, A is an attribute of r, and A' is a new attribute. Applying this operation forms a new relation r', where $attr(r') = (attr(r) - \{A\}) \cup \{A'\}$ and $|r'| = |\pi_{attr(r)-\{A\}} r|$. The value of A' for tuple t in r' is Y if and only if there exists a tuple t' in r such that $t'[attr(r) - \{A\}] = t[attr(r) - \{A\}]$ and $t'[A]$ is the literal string A', or is N if and only if there does not exist a tuple t' in r such that $t'[attr(r) - \{A\}] = t[attr(r) - \{A\}]$ and $t'[A]$ is the literal string A'.

- *Skolemization* φ. The φ operator has the form $\varphi_{f_A}(r)$, where f_A is a skolem function, and A is a new attribute. Applying this operation forms a new relation r', where $attr(r') = attr(r) \cup \{A\}$ and $|r'| = |r|$. The value of A for tuple t of row l in r' is a functional term that computes a value by applying the skolem function f_A over tuple t' of row l in r.

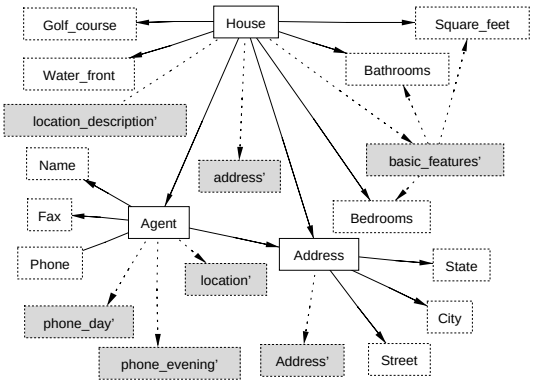
4 Example

As an example, let Schema 1 in Figure 1 be a target schema T of a data-integration system I. Assume a new information source becomes available for I, and let Schema 2 be a source schema S for the new information source after having been wrapped by a wrapper. A mapping tool in I generates a source-to-target mapping between S and T. Figure 2 illustrates the derivation over the source schema S and the virtual target-view elements in the source-to-target mapping. The shaded boxes denote derived object sets, and the dashed lines denote derived relationship sets. There are two main steps in the derivation.

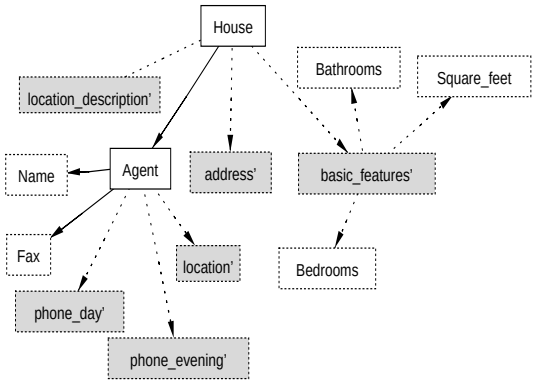
Step 1: *Use instance-level information to derive virtual source elements.* The implemented match-



(a) Result of Step 1



(b) Result of Step 2



(c) Virtual Target-View Elements

Figure 2: Derivation of Virtual Elements from Schema 2 for Schema 1

ing system applies expected-data-value techniques [EJX01] to derive object and relationship sets over S. Figure 2(a) shows the derived object and relationship sets after applying the following instance-level transformations.

- Derivation of $location\_description'$ and $House - location\_description'$.

$$House - location\_description' \Leftarrow \rho_{Golf\_course' \leftarrow location\_description'} \beta^{\text{“}Yes\text{”},\text{“}No\text{”}}_{Golf\_course, Golf\_course'}(House - Golf\_course) \cup \rho_{Water\_front' \leftarrow location\_description'} \beta^{\text{“}Yes\text{”},\text{“}No\text{”}}_{Water\_front, Water\_front'}(House - Water\_front)$$

$$location\_description' \Leftarrow \pi_{location\_description'}(House - location\_description')$$

- Derivation of $Address'$ and $Address - Address'$.

$$Address - Address' \Leftarrow \pi_{Address, Address'} \lambda_{(Street, \text{“ ”}, City, \text{“ ”}, State), Address'}(Address - Street \bowtie Address - City \bowtie Address - State)$$

$$Address' \Leftarrow \pi_{Address'}(Address - Address')$$

- Derivation of $phone\_day'$, $Agent-phone\_day'$, $phone\_evening'$, and $Agent-phone\_evening'$.[5]

$Agent - phone\_day' \Leftarrow \rho_{Phone \leftarrow phone\_day'} \sigma_{KEYWORD(day)}(Agent - Phone)$
$phone\_day' \Leftarrow \pi_{phone\_day'}(Agent - phone\_day')$
$Agent - phone\_evening' \Leftarrow \rho_{Phone \leftarrow phone\_evening'} \sigma_{KEYWORD(evening)}(Agent - Phone)$
$phone\_evening' \Leftarrow \pi_{phone\_evening'}(Agent - phone\_evening')$

Step 2: *Use schema-level information to derive virtual source elements.* The matching techniques apply source and target schema structural characteristics to derive object and relationship sets over S. Figure 2(b) shows the object and relationship sets in V_S after applying the following schema-level transformations.

- Derivation of $Agent - location'$, $location'$, $House - address'$, and $address'$.

$House - address' \Leftarrow \rho_{Address' \leftarrow address'} \pi_{House, Address'}(House - Address \bowtie Address - Address')$
$Agent - location' \Leftarrow \rho_{Address' \leftarrow location'} \pi_{Agent, Address'}(Agent - Address \bowtie Address - Address')$
$address' \Leftarrow \pi_{address'}(House - address')$
$location' \Leftarrow \pi_{location'}(Agent - location')$

- Derivation of $basic\_features'$, $House - basic\_features'$, $basic\_features' - Square\_feet$, $basic\_features' - Bedrooms$, and $basic\_features' - Bathrooms$.[6]

$House - basic\_features' \Leftarrow \varphi_{f_{basic\_features'}}(House)$
$basic\_features' - Bathrooms \Leftarrow \pi_{basic\_features', Bathrooms}(House - basic\_features' \bowtie House - Bathrooms)$
$basic\_features' - Bedrooms \Leftarrow \pi_{basic\_features', Bedrooms}(House - basic\_features' \bowtie House - Bedrooms)$
$basic\_features' - Square\_feet \Leftarrow \pi_{basic\_features', Square\_feet}(House - basic\_features' \bowtie House - Square\_feet)$

- Specializations of $Agent - phone\_day'$ and $Agent - phone\_evening'$.[7]

$Agent - phone\_day' \Leftarrow \sigma_{COMPATIBLE(agent-phone\_day)}(Agent - phone\_day')$
$Agent - phone\_evening' \Leftarrow \sigma_{COMPATIBLE(agent-phone\_evening)}(Agent - phone\_evening')$

[5] We may be able to recognize keywords such as *day-time*, *day*, *work phone*, *evening*, or *home* associated with each listed phone in the source. If so, we can apply the selection operator to sort out which phones belong in which set (if not, a human expert may not be able to sort these out either). We implement the $KEYWORD$ predicate by applying data-extraction techniques described in [ECJ+99].

[6] When applying the Skolemization operator to derive the virtual element $basic\_features'$, the system makes $basic\_features'$ functionally dependent on $House$ to match the functional dependency between $basic\_features$ and *house* in the target schema.

[7] The system specializes the relationship sets in the source so that they are compatible with the functional dependencies in the corresponding relationship sets in the target. The predicate $COMPATIBLE$ defaults to the first one or allows a user to decide how the selection should work. See [BE03] for a full explanation about source-target constraint incompatibilities.

At this point, Figure 2(c) is the subgraph of Figure 1(b) that contains exactly the virtual target-view elements in the mapping elements between S and T. For example, we have $(house \sim House)$, $(address \sim address')$, $(house - address \sim House - address')$,[8] and so forth. For each mapping element $(t \sim s \Leftarrow \theta_s(\Sigma_S))$ between S and T, the mediator of I derives an inclusion dependency $(S.s \subseteq t)$ for t. Thus, for each schema element t in a target schema $T \in I$, there is a set that contains all inclusion dependencies for t collected from all the information sources in I.

In summary, when a new information source becomes a part of the system I, a mapping tool exploits schema-matching techniques to semi-automatically generate a source-to-target mapping between the target schema T of I and a source schema for the new information source. Based on the source-to-target mapping, the mediator of I adjusts the inclusion dependencies for target relations in T automatically.

5 Query Reformulation

In the design phase, the data-integration system I collects the information including a target schema T, a set of source schemas $\{S_i\}$, and a set of source-to-target mappings $\{M_i\}$. In the query-processing phase, the system reformulates user queries in polynomial time.

To specify the semantics of I, we start with a *valid interpretation* D_{S_i} of a source schema $S_i \in \{S_i\} \in I$, $1 \leq i \leq n$. For an interpretation of a schema H to be valid, each tuple in D_H must satisfy the constraints specified for H. In our running example, let S_2 denote the information source whose schema is Schema 2 in Figure 1(b). For purposes of illustration, assume that we have one additional information source, which we denote as S_3, and assume that the wrapped Schema for S_3 is exactly the same as Schema 1 in Figure 1(a). Further, assume that both S_2 and S_3 have valid interpretations. The tables in Figure 3 show some partial populated data for relations in S_2

[8] Here, of course, we implicitly have the mapping expressions for $address'$ and $House - address'$, which we have just discussed.

House	Water_front
h1	Yes
h2	No
h3	Yes

House	Golf_course
h1	Yes
h2	Yes
h3	No

House	Square_feet
h1	990
h2	1420
h3	2500

(a) Partial Populated Data for Relations in S_2

house	location_description
h10	Water_front
h11	Golf_course

house	basic_features
h10	b10
h11	b11

basic_features	SQFT
b10	1000
b11	1500

(b) Partial Populated Data for Relations in S_3

Figure 3: Some Partial Populated Data for Relations in Sources

and S_3.

A target interpretation D_{S_iT} with respect to a source interpretation D_{S_i} in I (1) is a valid interpretation of T, and (2) satisfies the mapping M_i between S_i and T with respect to D_{S_i}. Assume that the mapping function for M_i is f_i. If f_i matches s_k with t_j, c is a tuple for t_j in D_{S_iT} if and only if c is a tuple for s_k derived through applying the mapping expression $\theta_{s_k}(\Sigma_{S_i})$ over D_{S_i}. For our running example, let the target schema be Schema 1 in Figure 1(a). Section 4 describes the source-to-target mapping between the target schema and the source schema for S_2 in Figure 1(b). Based on the source-to-target mapping, a valid target interpretation D_{S_2T} with respect to a valid interpretation D_{S_2} contains tuples in a valid interpretation for the schema in Figure 2(c). Moreover, since we assume that the source schema for S_3 is exactly the same as the target schema T, the source-to-target mapping between the target schema and the source schema for S_3 is trivial. Thus, the tuples in a valid target interpretation D_{S_3T} with respect to D_{S_3} are those of source relations in S_3.

The semantics of I, denoted as $sem(I)$, are defined as follows: $sem(I) = \{D_{S_iT} \mid D_{S_iT}$ is a target interpretation with respect to D_{S_i}, $S_i \in I\}$.[9] Intuitively, the semantics of I represent

[9]When sources share objects, both the object-identification problem and the data-merge problem need a resolution. (Note that neither this paper nor other papers that focus on data integration with virtual global schemas resolve

relevant data allowed in a predefined target schema T retrieved from available heterogeneous information sources. We are able to prove that if a source has a valid interpretation, then we can "load" data from the source into the target such that the part of the target populated from the source will necessarily have a valid interpretation [BE03].

In this paper, a query can be a conjunctive query, a conjunctive query with arithmetic comparisons, or a recursive query. We use logic-rule notation in [Ull88] to express user queries. Here, the queries are in terms of elements in Σ_T, which means that a predicate in a query body is either a target relation in Σ_T or a head predicate of a logic rule. We call a predicate representing a target relation appearing in a query body a *target predicate*. Like pure DataLog, we do not allow negations of predicates that appear in user queries because we adopt the "Open World Assumption" in our approach. In our running example, assume that a user wants an answer to the query, "For houses on water-front property, list the number of square feet." We can express this query, which we denote as $q_{example}$, using the following logic rule.

$$\begin{aligned} house-SQFT(x,y) \ :- \ & house-basic\_features(x,z) \ \& \ basic\_features-SQFT(z,y) \\ & \& \ house-location\_description(x, \text{“}Water\_front\text{”}) \end{aligned}$$

Let q be a user query such as the one above. When evaluating q over $sem(I)$, the system I transparently reformulates q as q^{Ext}, which is a query evaluated by retrieving data from the underlying information sources in I. Let $D = \{D_{S_i} | S_i \in \{S_i\} \in I\}$ be the set of valid interpretations of source schemas in I. By reformulating q as q^{Ext}, the system transforms the task of evaluating q over $sem(I)$ into a task of answering q^{Ext} over D.

The system I reformulates a user query q by applying the inclusion dependencies for target relations collected in the design phase. Since a user poses queries in terms of elements in Σ_T, each target relation r_i that appears in the body of q corresponds to a set of inclusion dependencies ID_i, $1 \leq i \leq N$ and $N = |\Sigma_T|$. We expand the user query q for each inclusion dependency $(S_j.s_k \ \subseteq \ r_i)$

these problems. The focus of this paper is on mediation, mappings, and query reformulation.)

in ID_i, where $s_k \in V_{S_j}$ ($1 \leq j \leq n$ and n is the number of sources), and where r_i is a target predicate that appears in the body of q, by adding a logic rule, $r_i(\overline{X}) \ :- \ S_j.s_k(\overline{X})$, where $\overline{X}$ is a vector of variables. Thus, the added logic rules as well as the logic rule of q together form the query q^{Ext}.

To reformulate $q_{example}$ for our running example, the system I applies inclusion dependencies for target relations $house - basic\_features$, $basic\_features - SQFT$, and $basic\_features - location\_description$ appearing in the body of query $q_{example}$. In addition to the logic rule above for the user query $q_{example}$, the following logic rules are added to form the reformulated query $q^{Ext}_{example}$.

$$house - basic\_features(x, z) \ :- \ S_2.House - basic\_features'(x, z)$$
$$house - basic\_features(x, z) \ :- \ S_3.house - basic\_features(x, z)$$
$$basic\_features - SQFT(z, y) \ :- \ S_2.basic\_features' - Square\_feet(z, y)$$
$$basic\_features - SQFT(z, y) \ :- \ S_3.basic\_features - SQFT(z, y)$$
$$house - location\_description(x, v) \ :- \ S_2.House - location\_description'(x, v)$$
$$house - location\_description(x, v) \ :- \ S_3.house - location\_description(x, v)$$

To evaluate a reformulated query q^{Ext} over sources, the system I decomposes q^{Ext} into sub-queries, and retrieves and combines query answers to sub-queries from individual information sources. We use a logic program P^{Ext}_D to describe the evaluation of q^{Ext} over D, where D is the set of valid interpretations of source schemas in I. The logic program P^{Ext}_D is defined as follows.

- *Rules.* The logic rules for q^{Ext}.
- *Facts.* For each source relation $S_j.s_k$ in the body of the logic program for q^{Ext}, we treat data for the source relations as ground facts. For example, if a tuple t is in the source relation $S_j.s_k$, we have the fact[10]: $S_j.s_k(t)$.

By evaluating P^{Ext}_D, the facts for the head predicate of q are query answers to the reformulated query q^{Ext} over D, which we denote as q^{Ext}_D. Note that when sending a sub-query to obtain data

[10]We use this logic program to capture the semantics of q^{Ext}_D. In real-world applications, query processing in I can optimize the evaluation of q^{Ext}.

from an information source S_j, the system I also sends the mapping expression $\theta_{s_k}(\Sigma_{S_j})$ to the source S_j so that the source S_j correctly executes the mapping expression to derive source facts for s_k. Given the data in Figure 3 for our running query example, $q_{example}$, we add the following list of source facts from S_2 and S_3 to the rules in $q^{Ext}_{example}$ above to form a logic program $P^{Ext}_{exampleD}$.

$S_2.House - location\_description'(h1, Water\_front)$
$S_2.House - location\_description'(h3, Water\_front)$
$S_2.House - location\_description'(h1, Golf\_course)$
$S_2.House - location\_description'(h2, Golf\_course)$
$S_2.House - basic\_features'(h1, f_{basic\_features'}(h1))$
$S_2.House - basic\_features'(h2, f_{basic\_features'}(h2))$
$S_2.House - basic\_features'(h3, f_{basic\_features'}(h3))$
$S_2.basic\_features' - Square\_feet(f_{basic\_features'}(h1), 990)$
$S_2.basic\_features' - Square\_feet(f_{basic\_features'}(h2), 1420)$
$S_2.basic\_features' - Square\_feet(f_{basic\_features'}(h3), 2500)$
$S_3.house - location\_description(h10, Water\_front)$
$S_3.house - location\_description(h11, Golf\_course)$
$S_3.house - basic\_features(h10, b10)$
$S_3.house - basic\_features(h11, b11)$
$S_3.basic\_features - SQFT(b10, 1000)$
$S_3.basic\_features - SQFT(b11, 1500)$

Note that the facts for S_2 have been transformed according to the source-to-target mapping in Section 4. By evaluating this logic program $P^{Ext}_{exampleD}$, we obtain the $house-SQFT$ facts $(h1, 990)$, $(h3, 2500)$, and $(h10, 1000)$ in $q^{Ext}_{exampleD}$.

With query reformulation in place, we can now prove that query answers to any query are *sound*—every answer to a user query is a fact according to the semantics of I—and *maximal*—the query answers contain all the facts the sources have to offer with respect to the facts allowed in the global target schema. Let q_I denote the query answers to a user query q over the semantics of I, $sem(I)$, which represents all data relevant to the target schema T from all information sources in I. The proofs are based on an observation that the semantics of q_I can be captured using a logic program P_I. The logic program P_I is defined as follows.

- *Rule.* A user query q in terms of target relations in T.
- *Facts.* For each tuple t for a target relation r in D_{S_jT}, where $D_{S_jT} \in sem(I)$ and $S_j \in \{S_j\} \in$

I, we have the fact: $r(t)$. (Note that these facts include all facts the sources have for T.)

The facts for the head predicate of q by evaluating P_I are the query answers q_I. Based on the characteristics of the two logic programs P_D^{Ext} and P_I, we can now prove the following theorem.

Theorem. *Let $I = (T, \{S_i\}, \{M_i\})$ be a data-integration system. Let $D = \{D_{S_i} | S_i \in \{S_i\} \in I\}$ be the set of valid interpretations of source schemas in I. Let q_I be the query answers obtains by evaluating q over $sem(I)$ and let q_D^{Ext} be the query answers obtained by evaluating q^{Ext} over D. Given a user query q in terms of target relations, a tuple $a = < a_1, a_2, \ldots, a_M >$ where M is the number of variables and constants in the head predicate of q, is in q_I if and only if a is a tuple in q_D^{Ext}.*

Proof (sketch). The two logic programs P_D^{Ext} and P_I respectively capture the semantics of q_D^{Ext} and q_I. We use resolution to show that a tuple $a = < a_1, a_2, \ldots, a_M >$ is a fact for the head predicate of query q by evaluating P_I if and only if the tuple a is a fact for the head predicate of query q by evaluating P_D^{Ext}. Based on the evaluation theory of logic programs in [SC90] and the definitions of P_D^{Ext} and P_I, we obtain the contradictions required by resolution. (See Appendix B for a detailed proof for this theorem.) □

6 Related Work

[CLL01] surveys the most important query processing algorithms proposed in the literature for LAV [LRO96, GKD97] and describes the principle GAV [CGMH+94] data-integration systems and the form of query processing they adopt. In a GAV approach, query reformulation reduces to rule unfolding. However, changes in information sources or adding a new information source requires a DBA to revise the global schema and the mappings between the global schema and source schemas. Thus, GAV is not scalable for large applications. LAV scales better and is easier to maintain than GAV because DBAs create a global schema independently of source schemas. Then, for a new (or

changed) source schema, the DBA only has to give (adjust) a source description that describes source relations as views of the global schema. Automating query reformulation in LAV, however, has exponential time complexity with respect to query and source schema definitions. Thus, LAV has low query performance when users frequently pose complex queries.

[FLM99] proposes a *Global-Local-as-View* (GLAV) approach, which combines the expressive power of both LAV and GAV. In a GLAV approach, the independence of a global schema, the maintenance to accommodate new sources, and the query-reformulation complexity are the same as in LAV. However, instead of using a restricted form of first-order logical sentences as in LAV and GAV to define view definitions, GLAV uses flexible first-order sentences such that it allows a view over source relations to be a view over global relations in source descriptions. Thus, GLAV can derive data using views over source relations, which is beyond the expressive ability of LAV, and it allows conjunctions of global relations, which is beyond the expressive ability of GAV. Our solution, TIQS, also has the ability to derive views over source schemas. The sets of view-creation operators in TIQS, however, are more powerful because GLAV has nothing comparable to merge/split or Boolean operators. Moreover, GLAV claims no ability to semi-automate the specification of source descriptions.

[CCGL02] proposes a translation algorithm to turn LAV into GAV such that it can keep LAV's scalability and obtain GAV's simple query reformulation. The translation results in a logic program that can be used to answer queries using rule unfolding. However, even though the translation to obtain the logic program is in polynomial time, the evaluation of the logic program could produce an exponential number of facts because of recomputing source relations over all source data. In contrast, TIQS encapsulates views for source relations in mapping elements. Since the view definitions are immediately available, query processing in TIQS has better query performance than the translation approach. Furthermore, [CCGL02] does not claim the ability to semi-automate

the specification of source descriptions.

The Clio project [MHH00, MHH+01] is a system for managing and facilitating the complex tasks of heterogeneous data transformation and integration. The objective of Clio is to support the generation and management of schemas, correspondences between schemas and mappings (queries) between schemas. Clio has an extensive tool set to aid users semi-automatically generate mappings. The system introduces an interactive mapping creation paradigm based on value correspondence that shows how a value of a target schema element can be created from a set of values of source elements. A DBA, however, is responsible to input most of the value correspondences. Clio and TIQS are independently implemented. The matching techniques in TIQS and Clio are complementary. The techniques of TIQS could help Clio discover both direct and indirect matches semi-automatically. On the other hand, TIQS could take advantage of the GUI provided by Clio such that a DBA can easily be involved in integrating data in an open environment. Currently, Clio claims to be useful for data warehousing. TIQS, however, provides an alternative approach for data integration without materializing data in global schemas. The components of either Clio and TIQS could be altered to support the alternative approach.

7 Conclusion

This paper describes TIQS, a unified, scalable approach to data integration using source-to-target mapping based on a predefined target schema, which combines the advantages and avoids the limitations of both GAV and LAV. Our solution has polynomial-time query reformulation and facilitates adding or modifying information sources in a data-integration system. DBAs create the target schema and wrap source schemas independently, so that neither the target schema nor the source schemas are contingent respectively on the source schemas or the target schema. TIQS has an implementation of a mapping tool that either creates or helps create the needed map-

pings, which reduces the heterogeneity between the source schemas and the target schema. Even when DBAs modify or add new items of interest to the target schema, the mapping tool of TIQS semi-automatically generates or adjust required source-to-target mappings between source schemas and the new target schema. Thus, TIQS increases both scalability and usability as compared to previously proposed approaches.

References

[BCV99] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, March 1999.

[BE03] J. Biskup and D.W. Embley. Extracting information from heterogeneous information sources using ontologically specified target views. *Information Systems*, 28(3):169–212, 2003.

[CCGL02] A. Cali, D. Calvanese, G. De Giacomo, and M. Lenzerini. On the expressive power of data integration systems. In *Proceedings of 21st International Conference on Conceptual Modeling (ER2002)*, pages 338–350, Tampere, Finland, October 2002.

[CGMH+94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of the 10th Meeting of the Information Processing Society of Japan*, pages 7–18, Tokyo, Japan, October 1994.

[CLL01] D. Calvanese, D. Lembo, and M. Lenzerini. Survey on methods for query rewriting and query answering using views. Technical report, University of Rome, Roma, Italy, April 2001.

[ECJ+99] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record Web pages. *Data & Knowledge Engineering*, 31(3):227–251, November 1999.

[EJX01] D.W. Embley, D. Jackman, and Li Xu. Multifaceted exploitation of metadata for attribute match discovery in information integration. In *Proceedings of the International Workshop on Information Integration on the Web (WIIW'01)*, pages 110–117, Rio de Janeiro, Brazil, April 2001.

[ETL02] D.W. Embley, C. Tao, and S.W. Liddle. Automatically extracting ontologically specified data from HTML tables with unknown structure. In *Proceedings of the 21st International Conference on Conceptual Modeling (ER2002)*, pages 322–337, Tampere, Finland, October 2002.

[FLM99] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI'99)*, pages 67–73, Orlando, Florida, 1999.

[GKD97] M.R. Genesereth, A.M. Keller, and O.M. Duschka. Infomaster: An information integration system. In *Proceedings of 1997 ACM SIGMOD International Conference on Management of Data*, pages 539–542, Tucson, Arizona, May 1997.

[Hal01] A.Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[LC00] W. Li and C. Clifton. SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering*, 33(1):49–84, 2000.

[LRO96] A.Y. Levy, A. Rajaraman, and J.J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB'96)*, pages 251–262, Mumbai (Bombay), India, September 1996.

[MHH00] R. Miller, L. Haas, and M.A. Hernandez. Schema mapping as query discovery. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB'00)*, pages 77–88, Cairo, Egypt, September 2000.

[MHH+01] R.J. Miller, M.A. Hernandez, L.M. Haas, L. Yan, C.T. Howard Ho, R. Fagin, and L. Popa. The clio project: Managing heterogeneity. *ACM SIGMOD Record*, 30(1):78–83, March 2001.

[Mil95] G.A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, November 1995.

[Qui93] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.

[RB01] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[SC90] L. Tanca S. Ceri, G. Gottlob, editor. *Logic Programming and Databases*. Springer-Verlag, Berlin Heidelberg, 1990.

[Ull88] J.D. Ullman. *Pinciples of Database and Knowledge-Base Systems, Vol. I.* Computer Science Press, New York, 1988.

[Ull97] J.D. Ullman. Information integration using logical views. In F.N. Afrati and P. Kolaitis, editors, *Proceedings of the 6th International Conference on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40, Delphi, Greece, January 1997.

[XE03] L. Xu and D.W. Embley. Discovering direct and indirect matches for schema elements. In *Proceedings of the 8th International Conference on Database Systems for Advanced Applications (DASFAA 2003)*, Kyoto, Japan, March 2003. (to appear).

A Examples for New Operators in the Mapping Algebra

A.1 Composition

Let r be the following relation, where $attr(r) = \{House,\ Street,\ City,\ State\}$.

House	Street	City	State
h1	339 Wymount Terrace	Provo	Utah
h2	15 S 900 E	Provo	Utah
h3	1175 Tiger Eye	Salt Lake City	Utah

Applying the operation $\lambda_{(Street,\text{“, ”},City,\text{“, ”},Street),Address}r$ yields a new relation r', where $attr(r') = \{House,\ Street,\ City,\ State,\ Address\}$.

House	Street	City	State	Address
h1	339 Wymount Terrace	Provo	Utah	339 Wymount Terrace, Provo, Utah
h2	15 S 900 E	Provo	Utah	15 S 900 E, Provo, Utah
h3	1175 Tiger Eye	Salt Lake City	Utah	1175 Tiger Eye, Salt Lake City, Utah

A.2 Decomposition

Let r be the following relation, where $attr(r) = \{House,\ Address\}$.

House	Address
h1	Provo, Utah
h2	339 Wymount Terrace, Provo, Utah
h3	1175 Tiger Eye, Salt Lake City, Utah

Applying the operation $\gamma^{R}_{Address,Street}r$, where R is a routine that obtains values of $Street$ from values of $Address$, yields a new relation r_1, where $attr(r_1) = \{House,\ Address,\ Street\}$.

House	Address	Street
h1	Provo, Utah	
h2	339 Wymount Terrace, Provo, Utah	339 Wymount Terrace
h3	1175 Tiger Eye, Salt Lake City, Utah	1175 Tiger Eye

Similarly, applying the operation $\gamma^{R'}_{Address,City}r$, where R' is a routine that obtains values of $City$ from values of $Address$, yields a new relation r_2, where $attr(r_2) = \{House,\ Address,\ City\}$.

House	Address	City
h1	Provo, Utah	Provo
h2	339 Wymount Terrace, Provo, Utah	Provo
h3	1175 Tiger Eye, Salt Lake City, Utah	Salt Lake City

A.3 Boolean

Let r be the following relation, where $attr(r) = \{House,\ Water\ Front\}$.

House	Water Front
h1	Yes
h2	No
h3	Yes

Applying the operation $\beta^{\text{“}Yes\text{”},\text{“}No\text{”}}_{Water\ Front,Lot\ Description}r$ yields a new relation r', where $attr(r') = \{House,\ Lot\ Description\}$.

House	Lot Description
h1	Water Front
h3	Water Front

A.4 DeBoolean

Let r be the following relation, where $attr(r) = \{House,\ Lot\ Description\}$.

House	Lot Description
h1	Water Front
h1	Golf Course
h1	Mountain View
h2	Water Front
h3	Golf Course

Applying the operation $\wr^{\text{“}Yes\text{”},\text{“}No\text{”}}_{Lot\ Description,Water\ Front}r$ yields a new relation r', where $attr(r') = \{House,\ Water\ Front\}$.

House	Water Front
h1	Yes
h2	Yes
h3	No

Similarly, applying the operation $\wr^{\text{“}x\text{”},\text{“”}}_{Lot\ Description,Golf\ Course}r$ yields a new relation r'', where $attr(r'')$ $= \{House,\ Golf\ Course\}$.

House	Golf Course
h1	x
h2	
h3	x

A.5 Skolemization

Let r be the following relation, where $attr(r) = \{House\}$.

House
h1
h2
h3

Applying the operation $\varphi_{f_{Basic\ Features}}r$ yields a new relation r', where $attr(r') = \{House,\ Basic\,Features\}$.

House	Basic Features
h1	$f_{Basic\ Features}(h1)$
h2	$f_{Basic\ Features}(h2)$
h3	$f_{Basic\ Features}(h3)$

B Theorem for Query Reformulation

Theorem. *Let $I = (T, \{S_i\}, \{M_i\})$ be a data-integration system. Let $D = \{D_{S_i} | S_i \in \{S_i\} \in I\}$ be the set of valid interpretations of source schemas in I. Let q_I be the query answers obtains by evaluating q over $sem(I)$ and let q_D^{Ext} be the query answers obtained by evaluating q^{Ext} over D. Given a user query q in terms of target relations, a tuple $a = < a_1, a_2, \ldots, a_M >$ where M is the number of variables and constants in the head predicate of q, is in q_I if and only if a is a tuple in q_D^{Ext}.*

Proof. Assume that we define two logic programs P_D^{Ext} and P_I based on the semantics of q_D^{Ext} and q_I respectively as we discussed in Section 5.

If. Assume that a tuple $a = < a_1, a_2, \ldots, a_M >$ is in q_D^{Ext} but not in q_I. Since a is in q_D^{Ext}, there exists a substitution ϑ, which binds variables in P_D^{Ext} with constants such that the evaluation of $P_D^{Ext}\vartheta$ yields the tuple a for the head predicate of q. By using a subset of substitution ϑ' of ϑ for variables in q while evaluating P_I, since a is not in q_I, there must exist at least one subgoal of q that is not satisfied in $P_I\vartheta'$. Based on the evaluation theory in [SC90], the subgoal could be a target predicate or an arithmetic comparison. We make an analysis for each possibility as follows.

- *Arithmetic comparison.* Assume that we have an arithmetic comparison, which is a subgoal of q, satisfied in $P_D^{Ext}\vartheta$ but not in $P_I\vartheta'$. Since we use the same bindings for variables of q in ϑ and ϑ', if the arithmetic comparison is satisfied in $P_D^{Ext}\vartheta$, it must also be satisfied in $P_I\vartheta'$. Thus, the unsatisfied subgoal is not an arithmetic comparison.

- *Target predicate.* Assume that we have a target predicate r, which is a subgoal of q, satisfied in $P_D^{Ext}\vartheta$ but not in $P_I\vartheta'$. Based on the substitution ϑ, the subgoal r is satisfied in P_D^{Ext} because $r(c)$ holds, where c is a vector of constants using the bindings in ϑ. Since $r(c)$ holds, there must exist a rule $r(c) \ :- \ S_j.s_k(c)$ and a fact $S_j.s_k(c)$, where $S_j \in \{S_i\} \in I$ and $s_k \in V_{S_j}$, in $P_D^{Ext}\vartheta$ such that $r(c)$ is derived from the rule and the fact. Since we define the rule, $r(c) \ :- \ S_j.s_k(c)$, based on an inclusion dependency $(S_j.s_k \ \subseteq \ r)$, there must exist a mapping element $(r \sim S_j.s_k \Leftarrow \theta_{s_k}(\Sigma_{S_j}))$ between S_j and T. Since s_k matches with r and c is a tuple of s_k in S_j, based on the semantics of D_{S_jT} in $sem(I)$, c is a tuple of r in D_{S_jT}. Then, based on the definition of P_I, there must exist a ground fact $r(c)$ in P_I. Since $r(c)$ is a ground fact in P_I, the subgoal $r(c)$ is satisfied in $P_I\vartheta'$. This is contrary to the assumption. Thus, the unsatisfied subgoal in P_I must not be a target predicate.

By analyzing the two possibilities, we conclude that all of the subgoals in q are satisfied while evaluating $P_I\vartheta'$ to obtain the tuple a as a fact for the head predicate of q. Thus, based on the semantics of P_I, a is a tuple in q_I. This is contrary to our assumption.

Only if. Assume that we have a tuple $a = < a_1, a_2, \ldots, a_M >$ in q_I but not in q_D^{Ext}. Since a is a tuple in q_I, there must exist a substitution ϑ for variables in q such that the evaluation of $P_I\vartheta$ outputs the tuple a as a fact for the head predicate of q. By using the same substitution ϑ for variables in q while evaluating P_D^{Ext}, since a is not in q_D^{Ext}, there must exist at least one subgoal in q that cannot be satisfied in $P_D^{Ext}\vartheta$. Based on the evaluation theory in [SC90], the subgoal could be a target predicate or an arithmetic comparison. We make an analysis for each possibility as follows.

- *Arithmetic comparison.* Assume that we have an arithmetic comparison, which is a subgoal of q, satisfied in $P_I\vartheta$ but not in $P_D^{Ext}\vartheta$. Since we use the same bindings for variables of q in

ϑ, if the arithmetic comparison is satisfied in $P_I\vartheta$, it must also be satisfied in $P_D^{Ext}\vartheta$. Thus, the unsatisfied subgoal is not an arithmetic comparison.

- *Target predicate.* Assume that we have a target predicate r, which is a subgoal of q, satisfied in $P_I\vartheta$ but not in $P_D^{Ext}\vartheta$. Since the subgoal r is satisfied in $P_I\vartheta$, $r(c)$ holds, where c is a vector of constants by using bindings in ϑ. Since $r(c)$ holds, based on the definition of P_I, c is a tuple of r in a target interpretation D_{S_jT} of $sem(I)$, where $S_j \in \{S_j\} \in I$. Based on the semantics of D_{S_jT}, since c is a tuple of r in D_{S_jT}, there must exist a mapping element $(r \sim S_j.s_k \Leftarrow \theta_{s_k}(\Sigma_{S_j}))$ between S_j and T where c is a tuple of the source relation $S_j.s_k$. By applying the mapping element that matches $S_j.s_k$ with r, we can derive an inclusion dependency $(S_j.s_k \subseteq r)$. Hence, since we have the inclusion dependency $(S_j.s_k \subseteq r)$ and the fact that c is a tuple of the source relation $S_j.s_k$, based on the definition of P_D^{Ext}, there must exist a rule $r(\overline{X}) :- S_j.s_k(\overline{X})$ and a fact $S_j.s_k(c)$ in $P_D^{Ext}\vartheta$, where $S_j \in \{S_j\} \in I$, $s_k \in V_{S_j}$, and $\overline{X}$ is a vector of variables corresponding attributes of r. Based on the rule and the fact, $r(c)$ holds in $P_D^{Ext}\vartheta$. This is contrary to the assumption. Thus, the unsatisfied subgoal in $P_D^{Ext}\vartheta$ must not be a target predicate.

By analyzing the two possibilities, we conclude that all of the subgoals in q are satisfied while evaluating $P_D^{Ext}\vartheta$ to obtain the tuple a as a fact for the head predicate of q. Thus, based on the semantics of P_D^{Ext}, a is a tuple in q_D^{Ext}. This is contrary to our assumption. □