Provenance Based Conflict Handling Strategies

Domenico Beneventano

Department of Computer Science
University of Modena and Reggio Emilia
Via Vignolese 905, 41125 Modena, Italy
domenico.beneventano@unimore.it

Abstract. A fundamental task in data integration is *data fusion*, the process of fusing multiple records representing the same real-world object into a consistent representation; data fusion involves the resolution of possible conflicts between data coming from different sources; several high level strategies to handle inconsistent data have been described and classified in [8].

The MOMIS Data Integration System [2] uses either *conflict avoiding* strategies (such as the *trust your friends* strategy which takes the value of a preferred source) and *resolution* strategies (such as the *meet in the middle* strategy which takes an average value).

In this paper we consider other strategies proposed in literature to handle inconsistent data and we discuss how they can be adopted and extended in the MOMIS Data Integration System. First of all, we consider the methods introduced by the *Trio* system [1,6] and based on the idea to tackle data conflicts by explicitly including information on provenance to represent uncertainty and use it to answer queries. Other possible strategies are to ignore conflicting values at the global level (i.e., only *consistent values* are considered) and to consider at the global level *all conflicting values*.

The original contribution of this paper is a provenance-based framework which includes all the above mentioned conflict handling strategies and use them as different *search strategies* for querying the integrated sources.

1 Introduction

A fundamental task in data integration is *data fusion*, the process of fusing multiple records representing the same real-world object into a consistent representation; data fusion involves the resolution of possible conflicts between data coming from different sources; several high level strategies to handle inconsistent data have been described and classified in [8].

MOMIS (Mediator envirOnment for Multiple Information Sources) is a framework to perform integration of structured and semi-structured data sources [3,2]. The MOMIS DAta Integration System is characterized by a classical wrapper/mediator architecture: the local data sources contain the real data, while a Global Schema (GS) provides a *reconciled, integrated, read-only view* of the underlying sources. The GS and the mapping between GS and the local sources have to be defined at design time by the Integration Designer; end-users can then pose queries over this GS. MOMIS has been

H. Yu et al. (Eds.): DASFAA Workshops 2012, LNCS 7240, pp. 286–297, 2012.

developed by the DBGROUP of the University of Modena and Reggio Emilia[1]. An open source version of the MOMIS system is delivered and maintained by the academic spin-off DataRiver[2].

The Data Fusion framework of the MOMIS System uses either *conflict avoiding* strategies (such as the *trust your friends* strategy which takes the value of a preferred source) and *resolution* strategies (such as the *meet in the middle* strategy which takes an average value); these strategies are implemented by means of the so-called *full outerjoin-merge operator* proposed in [16] and adapted to the MOMIS System in [2].

In this paper we consider other strategies proposed in literature to handle inconsistent data and we discuss how they can be adopted and extended in the MOMIS System. In particular, we consider the techniques introduced into the *Trio* system [1,6] and based on the idea to tackle data conflicts by explicitly including information on *lineage* to represent uncertainty and use it to answer queries.

Lineage, or *provenance*, in its most general definition, describes where data came from, how it was derived and how it was modified over time. Lineage provides valuable information that can be exploited for many purposes, ranging form simple statistical resumes presented to the end-user, to more complex applications such as managing data uncertainty or identifying and correcting data errors. For these reasons, in the last few years the research activity in the Information Management System area has been increasingly focused on this topic. In particular, lineage has been studied extensively in data warehouse systems [11,10]. However, in Data Integration systems, lineage is still considered as an open research problem [14,13].

In [4] we introduced the notion of provenance into the MOMIS framework, by defining the provenance for the *full outerjoin-merge operator*; this definition is based on the concept of *PI-CS*-provenance (Perm Influence Contribution Semantics) proposed in *Perm* [12] to produce more precise provenance information for outerjoins. Another important reason behind the choice to use the *PI-CS*-provenance is that it is implemented in an open-source provenance management system called *Perm* [12] (Provenance Extension of the Relational Model) that is capable of computing, storing and querying provenance for relational databases. We are using the *Perm* system as the SQL engine of the MOMIS system, so obtaining the provenance in our Data Integration System. On the other hand, from a theoretical point of view, in [4] we argued some differences between the *PI-CS*-provenance of the full outerjoin-merge operator and of the outerjoin operator and we are extending the *Perm* system to take into account these differences.

The original contribution of this paper is a provenance-based framework which includes several conflict handling strategies and use them as different *search strategies* for querying the Global Schema. Besides the techniques introduced into the *Trio* system [1,6], we will also take into account the strategy to ignore conflicting values at the global level, i.e., only *consistent values* are considered (in [8] this strategy is called *No Gossiping* and classified as a *conflict avoiding* strategy). Finally, another possible strategy, is to consider at the global level *all conflicting values* (in [8] this strategy is called *Considering all possibilities* and classified as a *conflict ignoring* strategy).

[1] `http://www.dbgroup.unimore.it`
[2] `http://www.datariver.it`

The remainder of the paper is organized as follows. In section 2 we will introduce the basic definitions of the MOMIS framework that will be used along the paper; section 2.1 briefly describes *conflict avoiding* and *resolution* strategies implemented in MOMIS by means of the *full outerjoin-merge operator*, whose provenance is introduced in section 2.2. The novel Provenance based Conflict Handling Strategies will be described in section 3 Finally, conclusions and future works are sketched in section 4.

2 The MOMIS Data Fusion System

In this section we will introduce the basic definition of the MOMIS framework [3,2] that will be used along this paper. A MOMIS Data Integration System is constituted by: a set of *local schemas* $\{LS_1, \ldots, LS_k\}$, a *global schema* GS and *Global-As-View* (*GAV*) mapping assertions [15] between GS and $\{LS_1, \ldots, LS_k\}$. A global schema GS is a set of *global classes*, denoted by G. A local schema LS is a set of *local classes*, denoted by L. Both the global and the local schemas are expressed in the ODL_{I^3} language [7]. However, for the scope of this paper, we consider both the GS and the LS_i as relational schemas, but we will refer to their elements respectively as global and local classes to comply with the MOMIS terminology.

For each global class G, a *Mapping Table* (*MT*) is defined, whose columns represent a set of local classes $\{L_1, \ldots, L_n\}$; an element $MT[GA][L]$ represents the local attribute of L which is mapped onto the global attribute GA, or $MT[GA][L]$ is empty (there is no local attribute of L mapped onto the global attribute GA)[3].

A small example, similar to the one used in [1], is shown in Figure 1; both local classes `LA` and `LB` contain data about crime vehicle sightings coming from two different data sources; Figure 1 also shows the Mapping Table of the global class `SAW`, with schema `SAW(Viewer,Age,Car)`, obtained by integrating these two local classes `LA` and `LB`.

GAV mapping assertions are expressed by specifying for each global class G a query over $\mathcal{L}(G)$, called *mapping query* and denoted by $\mathcal{MQ}^G$, which defines the instance of G starting from the instances of its local classes. The mapping query $\mathcal{MQ}^G$ is defined to make a global class perform *Data Fusion* among its local class instances [8]: multiple *local tuples* coming from local classes and representing the same real-world object are fused into a single and consistent *global tuple* of the global class. To identify multiple local tuples coming from local classes and representing the same real-world object, we assume that error-free and shared object identifiers exist among different sources: two local tuples with the same object identifier ID indicate the same object in different sources; thus we can use L^{id} to denote the tuple t of a local class L with object identifier ID equal to id, i.e. $t[ID] = id$. In our example, we assume `viewer` as an object identifier; then the first tuple of the local class `LA`, i.e. the tuple with `viewer` $= Amy$, will be denoted with $\texttt{LA}^{Amy}$.

[3] In this paper, for the sake of simplicity, we consider a simplified version of the MOMIS framework proposed in [3,2], where $MT[A][L]$ is a set of local attributes and *Data Transformation Functions* specify how local attribute values have to be transformed into corresponding global attribute values. Moreover we assume $S(G) = \cup_i S(L_i)$, i.e. global and local attribute names are the same.

Local classes LA and LB:

LA

viewer	age	car
Amy	21	Honda
Betty	NULL	Honda
Sam	NULL	Toyota
Joe	32	Toyota

LB

viewer	car
Amy	Toyota
Betty	Honda
Sam	Honda

Mapping Table of the global class Saw:

SAW	LA	LB
Viewer	viewer	viewer
Age	age	
Car	car	car

Fig. 1. Example: two local classes with a conflicting attribute

In fusing data from different sources into one consistent representation, several high level strategies have been described and classified in [8]. With respect to this classification, in MOMIS we used either *conflict avoiding* strategies (such as the *trust your friends* strategy which takes the value of a preferred source) and *resolution* strategies (such as the *meet in the middle* strategy which takes an average value); these solutions will be briefly described in section 2.1.

In this paper we consider how other strategies, proposed in literature, to handle inconsistent data, can be adopted and extended in the MOMIS Data Integration system. First of all, we consider the methods introduced into the *Trio* system [1,6] based on the idea to tackle data conflicts by explicitly including information on lineage to represents uncertainty and use it to answer queries. Another strategy is to ignore conflicting values at the global level, i.e., only *consistent values* are considered (in [8] this strategy is called *No Gossiping* and classified as a *conflict avoiding* strategy). Finally, another possible strategy, is to consider at the global level *all conflicting values* (in [8] this strategy is called *Considering all possibilities* and classified as a *conflict ignoring* strategy).

2.1 Data Fusion Strategies in MOMIS

In the MOMIS system, data fusion is performed at the level of a global class G by defining, for each conflicting attribute of G, a *Resolution Function* and by defining its mapping query $\mathcal{MQ}^G$ by means of the *full outerjoin-merge operator*, proposed in [16] and adapted to the MOMIS framework in [2]. Intuitively, this corresponds to the following two operations: (1) Computation of the *Full Outer Join*, on the basis of the shared object identifier, of the *local classes* of G; (2) Application of the resolution functions. Thus $\mathcal{MQ}^G$ can be formulated by standard SQL; for example, for the global class SAW:

```
MQ^SAW : SELECT viewer AS Viewer, age AS Age,
                COALESCE(SAW_B.car,SAW_A.car) AS Car
         FROM LA FULL OUTER JOIN LB
                              USING (viewer)
```

where COALESCE is the standard SQL function which returns its first non-null parameter value. The obtained instance of the global class SAW is shown in Table 1. The global attribute Viewer, derived from the shared object identifier viewer, is an object identifier for the global class SAW; the global tuple with Viewer $= Amy$, is denoted with $\texttt{SAW}^{Amy}$.

Table 1. Instance of the global class SAW

Viewer	Age	Car
Amy	21	Toyota
Betty	NULL	Honda
Sam	NULL	Honda
Joe	32	Toyota

2.2 Provenance in the MOMIS System

In [4] we defined the provenance for the full outerjoin-merge operator by using the concept of *PI-CS*-Provenance. The *PI-CS*-Provenance of an output tuple is a set of *witness lists*, where each witness list represents one combination of input relation tuples that were used together to derive the output tuple; a witness list contains a tuple from each input of an operator or the special value $\perp$ which indicates that no tuple from an input relation was used to derive the output tuple and, therefore, is useful in modeling outerjoins. For example, the *PI-CS*-Provenance of the global tuple $\texttt{SAW}^{Amy}$ is the set $\{\langle LA^{Amy}, LB^{Amy} \rangle\}$ and the *PI-CS*-Provenance of $\texttt{SAW}^{Joe}$ is the set $\{\langle LA^{Joe}, \perp \rangle\}$ which indicates that $\texttt{LA}^{Joe}$ paired with no tuples from LB influences $\texttt{SAW}^{Joe}$.

Another important reason behind the choice to use *PI-CS*-Provenance is that it is implemented in a provenance management system called *Perm* [12] (Provenance Extension of the Relational Model) that is capable of computing, storing and querying provenance for relational databases. From an implementation point of view, we are using the *Perm* system as the SQL engine of the MOMIS system, so obtaining the provenance in our Data Integration System. On the other hand, from a theoretical point of view, in [4] we argued some differences between the *PI-CS*-provenance of the full outerjoin-merge operator and of the outer join operator; however, this aspect is not relevant for the scope of this paper.

As an example of provenance, let us consider the following query on the global class SAW (the user is searching for the suspected cars defined as the car sighted by viewers with age greater than 18):

```
SUSPECTED_CAR = SELECT DISTINCT CAR
                FROM SAW
                WHERE AGE > 18
```

The query returns the tuple $(Toyota)$; the *PI-CS*-Provenance for this tuple is a set with two witness lists (see Figure 2). In the *Perm* system witness lists are represented in a relational form, as shown in Figure 2: each witness list of a result tuple is represented by a single tuple. This is the data provenance information obtained by executing the query on the MOMIS system integrated with the *Perm* system.

Car	*PI-CS* Provenance as a set of witness lists
Toyota	$\{ \langle \texttt{LA}^{Amy}, \texttt{LB}^{Amy} \rangle, \langle \texttt{LA}^{Joe}, \perp \rangle \}$

Relational Representation of the *PI-CS* Provenance

Car	LA.viewer	LA.age	LA.car	LB.viewer	LB.car
Toyota	Amy	21	Honda	Amy	Toyota
Toyota	Joe	32	Toyota		

Fig. 2. Example: *PI-CS* Provenance for the query SUSPECTED_CAR

3 Provenance Based Conflict Handling Strategies

The proposed solution to handle inconsistent data is inspired to the methods introduced into the *Trio* system [1,6], which is based on the *ULDB* model [5], where *uncertainty* and *lineage* of the data are considered as first-class concepts. The idea is to tackle data conflicts by explicitly including information on lineage (and accuracy) into its data model and thereby into the data itself; as stated in [5], any application that integrates information from multiple sources may be uncertain about which data is correct, and the original source and derivation of data may offer helpful additional information.

With the concept of *alternatives* introduced in *ULDB*, relations have a set of *certain* attributes and a set of *uncertain* attributes; each tuple in a *ULDB* relation has one value for each certain attribute, and a set of possible values for the uncertain attributes. The *ULDB* model is based on the idea that *provenance* enables simple and consistent representation of uncertain data; provenance in *ULDB* is recorded at the granularity of alternatives: provenance connects a tuple-alternative to those tuple-alternatives from which it was derived. Specifically, provenance is defined as a function λ over tuple-alternatives: $\lambda(t)$ is a boolean formula over the tuple-alternatives from which the alternative t was derived. In the following, we will refer to the provenance defined in *ULDB* as *Trio*-Lineage.

Intuitively, by applying this concept of *alternatives* to our context of data fusion, *non-conflicting* attributes are modelled as *certain* attributes and *conflicting* attributes are modelled as *uncertain* attributes where conflicting values are considered as *mutually exclusive* values for the global tuple; in this way, the global class SAW is considered as an *uncertain relation* which represents a set of possible relation instances. We will refer to this conflict handling strategy as the *TRIO*-strategy. In our example (see Table 2), with the *TRIO*-strategy we have *four possible instances* for the global class SAW: two choices for Amy's car times two choices for Sam's car. We use G^{id} to denote the global tuple of G with object identifier ID equal to id, then we will use (G^{id}, j) to denote the *j*th tuple-alternative of the global tuple G^{id}; as an example, for the global tuple $\texttt{SAW}^{Amy}$ there are two alternatives, $(\texttt{SAW}^{Amy}, 1)$ and $(\texttt{SAW}^{Amy}, 2)$. Given a global class constituted by n local classes and with only a conflicting attribute GA (the general case with more than one conflicting attribute will be considered in section 3.2) mapped on k local classes $L_1, \ldots, L_k$, with $1 < k \leq n$, the number of tuple-alternatives for a global tuple is of course equal or less than k.

Table 2. Global class SAW as an *uncertain* relation: global tuple-alternatives

Viewer	Age	Car
Amy	21	Honda || Toyota
Betty	NULL	Honda
Sam	NULL	Toyota || Honda
Joe	32	Toyota

For a global class, the provenance connects a *global tuple* to local tuples from which it was derived; by using the *ULDB* to model a global class as an uncertain relation, *Trio*-Lineage connects a *global tuple-alternative* to those local tuples from which it was derived. More precisely, the *Trio*-Lineage of a global tuple-alternative is a boolean formula over the local tuples from which the alternative was derived; this is shown in Table 3 where each *global tuple-alternative* of SAW is represented by a row of the table with the related *TRIO*-Lineage.

Table 3. Global tuple-alternatives of SAW, with the related *TRIO*-Lineage

Viewer	Age	Car	*TRIO*-Lineage
Amy	21	Honda	$\lambda(\texttt{SAW}^{Amy}, 1) = \texttt{LA}^{Amy}$
Amy	21	Toyota	$\lambda(\texttt{SAW}^{Amy}, 2) = \texttt{LA}^{Amy} \wedge \texttt{LB}^{Amy}$
Betty	NULL	Honda	$\lambda(\texttt{SAW}^{Betty}, 1) = \texttt{LA}^{Betty} \vee \texttt{LB}^{Betty}$
Sam	NULL	Toyota	$\lambda(\texttt{SAW}^{Sam}, 1) = \texttt{LA}^{Sam}$
Sam	NULL	Honda	$\lambda(\texttt{SAW}^{Sam}, 2) = \texttt{LB}^{Sam}$
Joe	32	Toyota	$\lambda(\texttt{SAW}^{Joe}, 1) = \texttt{LB}^{Joe}$

With the *TRIO*-strategy a global class is considered as an *uncertain relation* which represents a set of possible instances. Another possible strategy is to consider *a unique instance* containing *all conflicting values*, i.e., an instance with all the global tuple-alternatives shown in Table 3; we will refer to this strategy as the *ALL* strategy. Finally, another possible strategy is to ignore conflicting values at the global level, i.e., only *consistent values* are considered; this strategy called *No Gossiping* and classified as an *avoiding* strategy in [8], is similar to the concept of *Consistent Query Answering* [9]. With this strategy, which we will call *CQA* strategy, there is a unique instance for the global class SAW : $\{\texttt{SAW}^{Betty}, \texttt{SAW}^{Joe}\}$.

The original contribution of this paper is a provenance-based framework which includes all the above conflict handling strategies. In the next section, we will discuss how these different conflict handling strategies correspond to different *search strategies* for querying the Global Schema. In the remaining of this section we show how to implement the proposed framework.

Since in the MOMIS System we are already using the *Perm* system as the basis to obtain the provenance of the integrated data, the proposal is to implement the framework using the *Perm* system. This choice is motivated by several reasons. Firstly, besides the *Trio*-strategy, we want to obtain a framework which also contain the other two kind of

strategies, then extensions to the *Trio* framework are needed. Moreover, as shown in section 2.1, the computation of the global class is based on a outerjoin operation, then we need a provenance model where this operation is supported. In the *ULDB* model [5] and in the *Trio* system [1,6] outerjoins are not considered.

In the following we will illustrate how to compute in the MOMIS system the Global tuple-alternatives of a global class, with the related *TRIO*-Lineage (see the example of SAW in Table 3). First of all, a relational representation of the *Trio*-Lineage is intuitively obtained as follows; the *Trio*-Lineage of a global tuple-alternative is a boolean formula over the local tuples; we consider the disjunctive normal form of this formula and each conjunctive clause of this formula is represented by a row; for example, since $\lambda(\texttt{SAW}^{Betty}, 1) = \texttt{LA}^{Betty} \vee \texttt{LB}^{Betty}$ we will obtain two rows for the alternative $(\texttt{SAW}^{Betty}, 1)$, the first one representing $\texttt{LA}^{Betty}$ and the second one representing $\texttt{LB}^{Betty}$ (see Table 4). For a global class G, the instance G^{all} containing all the global

Table 4. Relational Representation of the *Trio*-Lineage for SAW

Viewer	Age	Car	LA.viewer	LA.age	LA.car	LB.viewer	LB.car
Amy	21	Honda	Amy	21	Honda		
Amy	21	Toyota	Amy	21	Honda	Amy	Toyota
Betty	NULL	Honda	Betty	NULL	Honda		
Betty	NULL	Honda				Betty	Honda
Joe	32	Toyota				Joe	Toyota
Sam	NULL	Toyota	Sam	NULL	Toyota		
Sam	NULL	Honda				Sam	Honda

tuple-alternatives of G, with their relational representation of the *Trio*-Lineage, is obtained as follows:

- non-conflicting attributes $GA_1, \ldots, GA_n$ are computed by the full outerjoin-merge operator defined in section 2.1 (see MQ^SAW); we denote by $G1(ID, GA_1, \ldots, GA_n)$ the obtained relation (where ID is the object identifier);
- the conflicting attribute GA_{n+1} is computed by the union of k queries on the local classes $L_1, \ldots, L_k$ where it is mapped; we denote by $G2(ID, GA_{1+1})$ the obtained relation; as an example, for the conflicting attribute Car of SAW:

```
G2 : SELECT viewer, car from  LA
     UNION
     SELECT viewer, car from  LB
```

The relation G^{all} is then obtained as the natural full outerjoin between G1 and G2; the result for the global class SAW is shown in Table 4, where global tuple with more than one alternative are highlighted.

3.1 Strategies for Querying the Global Schema

In this section we discuss how the different conflict handling strategies introduced in the previous section can be used as different *search strategies*: the same query on the

Global Schema has a different interpretation on the basis of the chosen conflict handling strategy. In particular, with the *TRIO*-strategy, the *membership problems* of uncertain databases are considered: *Tuple Membership*: Is a given tuple in some possible instance of an uncertain relation?
Tuple Certainty: Is a given tuple in every instance of an uncertain relation?
In [17] is shown that the tuple membership and certainty problems can be solved in polynomial time and algorithms are given.

Given a query Q, in the following we will define (in an informal way) and compare the query answers obtained with the different conflict handling strategies (as an example we consider the SUSPECTED_CAR of page 2.2).

***CQA*-strategy:** $Q^{consistent}$ is the answer obtained by considering only *consistent values*; in the example: $\texttt{SUSPECTED\_CAR}^{consistent} = \{Toyota\}$.

***ALL*-strategy:** Q^{all} is the answer obtained by considering *all values*;
in the example: $\texttt{SUSPECTED\_CAR}^{all} = \{Honda, Toyota\}$.

***TRIO*-strategy:** $Q^{possible}$ is the set of all *possible* tuples of the uncertain relation Q, i.e., is the set of tuples which are in some possible instance of Q (*Tuple Membership*);
in the example: $\texttt{SUSPECTED\_CAR}^{possible} = \{Honda, Toyota\}$.

***TRIO*-strategy:** $Q^{certain}$ is the set of all *certain* tuples of the uncertain relation Q, i.e., is the set of tuples which are in all possible instances of Q (*Tuple Certainty*);
in the example: $\texttt{SUSPECTED\_CAR}^{certain} = \{Toyota\}$.

It easy to verify that the following relationships hold among these query answers:

$$Q^{consistent} \subseteq Q^{certain} \subseteq Q^{possible} \subseteq Q^{all}$$

On the basis of these relationships, the *CQA* strategy is the *most restrictive* search strategy (i.e., brings back fewer results) and the *ALL* strategy is the *least restrictive* one.

Table 5 shows Q^{all} and, for each tuple of this result, the *most restrictive* search strategy which brings back the tuple: in this way the user knows that $Toyota$ can be obtained from *consistent values* while to obtain $Honda$ also *conflicting values* need to be considered. Then the user can ask for information about the provenance, by obtaining the result shown in Table 6: in this way the user knows that $Toyota$ has two derivations, the first one from *consistent values* and the second one from a possible instance, i.e., from *conflicting values*.

Table 5. All tuples in SUSPECTED_CAR with the related *most restrictive* search strategy

Car	Strategy
Toyota	consistent
Honda	possible

3.2 Dependent and Independent Conflicting Attributes

So far we considered only one conflicting attribute; with two ore more conflicting attributes, another dimension need to be take into account for the conflict handling problem; with reference to Figure 3 the question is whether to consider the tuple

Table 6. Relational Representation of the *Trio* Lineage for SUSPECTED_CAR

Car	**Strategy**	LA.viewer	LA.age	LA.car	LB.viewer	LB.car
Toyota	consistent	Joe	32	Toyota		
Toyota	possible	Amy	21	Honda	Amy	Toyota
Honda	possible	Amy	21	Honda		

$(Amy, 21, Honda, brown)$ as a possible instance of the global class SAW? The answer is affirmative if we consider the two conflicting attributes Color and Car as *independents*: Car is taken from local class LA and Color is taken from local class LB. The answer is negative if we consider the two conflicting attributes as *dependents*: Color and Car are taken from the same local class. As shown in Figure 3, if we consider the conflicting attributes as *independents*:

- for the global tuple $\texttt{SAW}^{Amy}$ there are two further alternatives
- for the global tuple $\texttt{SAW}^{Betty}$ there are no further alternatives but the alternative with $(Honda, red)$ has now two different derivation, i.e. its *TRIO*-Lineage is $\lambda(\texttt{SAW}^{Betty}, 1) = \texttt{LA}^{Betty} \vee (\texttt{LA}^{Betty} \wedge \texttt{LB}^{Betty})$.

LA

viewer	age	car	color
Amy	21	Honda	red
Betty	NULL	Honda	red
Sam	NULL	Toyota	red
Joe	32	Toyota	brown

LB

viewer	car	color
Amy	Toyota	brown
Betty	Honda	brown
Sam	Honda	red

Global class SAW as an *uncertain* relation: global tuple-alternatives

Viewer	Age	(Car,Color)
Amy	21	(Honda,red) || (Toyota, brown) || (Honda,brown) || (Toyota, red)
Betty	NULL	(Honda,red) || (Honda, brown)
Sam	NULL	(Toyota, red) || (Honda,red)
Joe	32	(Toyota, brown)

Fig. 3. Example: two local classes with two conflicting attributes

Thus, we can consider different instances for a global class with two (or more) conflicting attributes: an instance obtained by considering *dependent* attributes is, of course, a subset of the one obtained by considering the attributes as *independents*. As a consequence, given a query Q on a global class, for each conflict handling strategies S (except for the *CQA* strategy), we can consider two query answers : Q^S_{depend} and $Q^S_{independ}$, with $Q^S_{depend} \subseteq Q^S_{independ}$. The *CQA* strategy is the *most restrictive* search strategy and the *ALL* strategy with *independent* attributes is the *least restrictive* one. To give an example, let us refine the SUSPECTED_CAR query as follows:

```
SUSPECTED_RED_CAR = SELECT DISTINCT CAR
                    FROM G
                    WHERE AGE > 18
                    AND COLOR = 'RED'
```

For this query, we have the following search strategies:

- *most restrictive*: $\texttt{SUSPECTED\_RED\_CAR}^{consistent} = \emptyset$
- *least restrictive* with dependent attributes: $\texttt{SUSPECTED\_RED\_CAR}^{all}_{dep} = \{Toyota\}$
- *least restrictive* with independent attributes: $\texttt{SUSPECTED\_RED\_CAR}^{all}_{indep} = \{Honda, Toyota\}$

Table 7 shows $\texttt{SUSPECTED\_RED\_CAR}^{all}_{indep}$ and, for each tuple of this result, the *most restrictive* search strategy which brings back the tuple: in this way the user knows that to obtain $Honda$ and $Toyota$ *conflicting values* need to be considered. and that $Toyota$ can be obtained only considering *independent* attributes. The provenance is shown in Table 8.

Table 7. SUSPECTED_RED_CAR

Car	Strategy
Honda	possible-dependent
Toyota	possible-independent

Table 8. Relational Representation of the *Trio* Lineage for SUSPECTED_RED_CAR

Car	Strategy	LA.viewer	LA.age	LA.car	LA.color	LB.viewer	LB.car	LB.color
Honda	possible dependent	Amy	21	Honda	red			
Toyota	possible independent	Amy	21	Honda	red	Amy	Toyota	blue

4 Conclusion and Future Work

In this paper we presented our work in progress to implement data provenance techniques in the MOMIS Data Integration System; currently, the provenance is obtained by using as SQL engine of our Data Integration System the open-source provenance management system *Perm* [12]. The original contribution of this paper are some idea to develop a provenance-based framework which enables different conflict handling strategies and use them as different search strategies for querying the Global Schema. In particular, we considered the provenance-based techniques to tackle data conflicts introduced into the *Trio* system [1,6] and we discussed how they can be adopted, extended and implemented in our system.

As future works, from a theoretical point of view, a formal account for the introduced ideas will be given. From an implementation point of view, since also the *Trio* source code is freely available and the system is based also on PostgreSQL, another possible

implementation is to use the*Trio* as SQL engine of our Data Integration System and then evaluate which extensions are necessary (such as the outerjoins) to realize the proposed framework.

References

1. Agrawal, P., Benjelloun, O., Sarma, A.D., Hayworth, C., Nabar, S., Sugihara, T., Widom, J.: Trio: a system for data, uncertainty, and lineage. In: VLDB 2006: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 1151–1154. VLDB Endowment (2006)
2. Beneventano, D., Bergamaschi, S., Guerra, F., Orsini, M.: Data integration. In: Embley, D., Thalheim, B. (eds.) Handbook of Conceptual Modelling. Springer, Heidelberg (2010), http://dbgroup.unimo.it/SSE/SSE.pdf
3. Beneventano, D., Bergamaschi, S., Guerra, F., Vincini, M.: Synthesizing an integrated ontology. IEEE Internet Computing 7(5), 42–51 (2003)
4. Beneventano, D., Dannoui, A.R., Sala, A.: Data lineage in the momis data fusion system. In: ICDE Workshops of the 27th International Conference on Data Engineering, ICDE 2011, Hannover, Germany, April 11-16, pp. 53–58 (2011)
5. Benjelloun, O., Sarma, A.D., Halevy, A., Widom, J.: Uldbs: databases with uncertainty and lineage. In: VLDB 2006: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 953–964. VLDB Endowment (2006)
6. Benjelloun, O., Sarma, A.D., Hayworth, C., Widom, J.: An introduction to uldbs and the trio system. IEEE Data Eng. Bull. 29(1), 5–16 (2006)
7. Bergamaschi, S., Castano, S., Vincini, M., Beneventano, D.: Semantic integration of heterogeneous information sources. Data Knowl. Eng. 36(3), 215–249 (2001)
8. Bleiholder, J., Naumann, F.: Data fusion. ACM Comput. Surv. 41(1), 1–41 (2008)
9. Chomicki, J.: Consistent Query Answering: Five Easy Pieces. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 1–17. Springer, Heidelberg (2006)
10. Cui, Y., Widom, J.: Lineage tracing for general data warehouse transformations. The VLDB Journal 12(1), 41–58 (2003)
11. Cui, Y., Widom, J., Wiener, J.L.: Tracing the lineage of view data in a warehousing environment. ACM Trans. Database Syst. 25(2), 179–227 (2000)
12. Glavic, B., Alonso, G.: Perm: Processing provenance and data on the same data model through query rewriting. In: Proceedings of the 2009 IEEE International Conference on Data Engineering, pp. 174–185. IEEE Computer Society, Washington, DC (2009)
13. Halevy, A., Li, C.: Information integration research: Summary of nsf idm workshop breakout session. In: NSF IDM Workshop (2003)
14. Halevy, A., Rajaraman, A., Ordille, J.: Data integration: the teenage years. In: VLDB 2006: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 9–16. VLDB Endowment (2006)
15. Lenzerini, M.: Data integration: A theoretical perspective. In: PODS, pp. 233–246 (2002)
16. Naumann, F., Freytag, J.C., Leser, U.: Completeness of integrated information sources. Inf. Syst. 29(7), 583–615 (2004)
17. Sarma, A.D., Benjelloun, O., Halevy, A., Nabar, S., Widom, J.: Representing uncertain data: models, properties, and algorithms. The VLDB Journal 18(5), 989–1019 (2009)