Data Lineage in the MOMIS Data Fusion System

Domenico Beneventano #1, Abdul Rahman Dannoui #2, Antonio Sala #3

#Dipartimento di Ingegneria dell’Informazione
Università di Modena e Reggio Emilia
Via Vignolese 905, 41125 Modena, Italy
1domenico.beneventano@unimore.it
2abdulrahman.dannaoui@unimore.it
3antonio.sala@unimore.it

Abstract—Data Lineage is an open research problem. This is particularly true in data integration systems, where information coming from different sources, potentially uncertain or even inconsistent with each other, is integrated. In this context, having the possibility to trace the lineage of certain data can help unraveling possible unexpected or questionable results.

In this paper, we describe our preliminary work about this problem in the context of the MOMIS data Integration system. We discuss and compare the use of Lineage-CS and PI-CS provenance, introduced respectively in [1] and [2], for the data fusion operator used in the MOMIS system; in particular we evaluate how the computation of the PI-CS provenance should be extended to deal with Resolution Functions used in our data fusion system.

I. INTRODUCTION

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 from 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 [1], [3]. However, in Data Integration systems, lineage is still considered as an open research problem [4], [5]. Data Integration systems deal with information coming from different sources, potentially uncertain or even inconsistent with each other. In this context, collecting lineage information becomes a necessity. Lineage information helps the integration process by improving the system capability to introspect about the sources reliability and the certainty of the data.

A fundamental task in data integration is data fusion, the process of fusing multiple records representing the same real-world object into a single, consistent, and clean representation. Data fusion involves the resolution of possible conflicts between data coming from different sources [6]. A recent tutorial [7] listed data lineage as one of the open problems and desiderata for data fusion systems. Merging data implies a partial loss of the original values of the local sources. For this reason database administrators and data owners are notoriously hesitant to merge data. Data lineage can help explaining merging decisions by tracking which original values were involved and how they have been fused together.

In this paper, we describe our preliminary work on the application of data lineage techniques in the Data Fusion framework of the MOMIS Integration System. As described in previous works [8], [9], MOMIS (Mediator envirOnment for Multiple Information Sources) is a framework to perform integration of structured and semi-structured data sources. MOMIS 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, together with Resolution Functions to solve data conflicts. End-users can then pose queries over this GS.

As observed in [10], two types of information are in general considered to form the provenance of a data item. Information about the source data items that contributed to a result and the transformations used in the creation process of the data item. Source provenance can be distinguished by the definition of “contribution” used to estimate if an input data item influenced a result data item. Buneman et al. [11] described two definitions of contribution semantics: Why- and Where-provenance. Why-Provenance includes all data items that influenced the creation of a data item. Where-Provenance includes only parts of the input that we copied literally to the output.

This paper focuses on Why-Provenance, to describe the input tuples of a query q that contributed to an output tuple of q. The maximal set of tuples from source tables that produced an output tuple is usually identified [1]. Recently, in TRAMP [2], in contrast to this interpretation (which we will refer to as Lineage-CS as proposed in [2]), the concept of PI-CS (Perm Influence Contribution Semantics) is introduced to produce more precise provenance information for outer joins and union. The concept of PI-CS provenance seems more suitable for a data fusion framework where outer join and union are fundamental operators.

In this paper, we discuss and compare the use of Lineage-CS and PI-CS provenance for the data fusion operator used in the MOMIS system; in particular we evaluate how the computation of the PI-CS provenance should be extended to
deal with Resolution Functions used in our data fusion system.

The remainder of the paper is organized as follows. In section 2 we will introduce the basic definition of the MOMIS framework that will be used along the paper. In section III we first recall the definition of Lineage-CS and PI-CS Provenance given in [2], then we apply and compare these definitions to the data fusion operator of MOMIS. Conclusion and Future Work are in Section IV.

II. THE MOMIS DATA FUSION SYSTEM

In this section we will introduce the basic definition of the MOMIS framework [8], [9] that will be used along this paper. MOMIS has been developed by the DBGROUP of the University of Modena and Reggio Emilia. An open source version of the MOMIS system is delivered and maintained by the academic spin-off DataRiver.

A MOMIS Data Integration System is constituted by: a set of local sources with their related local schemas, a global schema (GS) and Global-As-View (GAV) mapping assertions [13] between GS and the local schemas. For each global class G, a Mapping Table (MT) is defined, whose columns represent the local classes belonging to G. For each of its columns which represents the local attributes of G, 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). A global attribute GA such as there is only a not null element MT[GA][L] is called one-to-one, otherwise is called one-to-many.

As an example, starting from the following three local classes (relations):

L1 (ID, A, C), L2 (ID, B, C), L3 (ID, C)

the global class G and the related Mapping Table of Figure 1 are automatically generated.

A Global Class performs Data Fusion among its local class instances [6]: multiple records coming from local classes and representing the same real-world object are fused into a single and consistent record of the global class. To identify multiple 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 records with the same object identifier indicate the same object in different sources. In our example, we assume ID as an object identifier.

Data Reconciliation, i.e. to solve conflicts among instantiations of the same object in different sources, is performed by Resolution Functions [14]: for each GA such that there are more than one non empty element MT[GA][L], a Resolution Function (RF) is defined to obtain, starting from the transformed local classes, a merged value for GA. In our example an AVG resolution function is defined on C.

Finally, GAV mapping assertions are expressed by defining for each global class G a mapping query MQG by means of the full outerjoin-merge operator proposed in [15] and adapted to the MOMIS framework in [9]; intuitively, it corresponds to the following two operations: (1) Computation of the Full Outer Join, on the basis of the shared object identifiers, of the local classes; (2) Application of the Resolution Functions. A mapping query can be rewritten by standard SQL, with the exception of (some) resolution functions. For the global class G of Figure 1 we have the following MQG:

\[
MQ_G : \text{SELECT} \quad \text{COALESCE}(L1.ID,L2.ID,L3.ID) \text{ AS ID,} \quad L1.A \text{ AS A,} \quad L2.B \text{ AS B,} \quad \text{AVG}(L1.C,L2.C,L3.C) \text{ AS C} \\
\text{FROM} \quad \text{L1 FOJ L2 ON} \quad (L2.ID=L1.ID) \quad \text{FOJ L3 ON} \quad (L3.ID=L1.ID \text{ OR L3.ID=L2.ID})
\]

where FOJ is an abbreviation for the SQL full outer join operator. COALESCE is the standard SQL function which returns its first non-null parameter value and AVG is a (non standard SQL) function to compute the average value.

In other words, MQG is a relational operator with inputs \( L_1, \ldots, L_n \); the instances of \( G \) are computed as MQG, i.e., \( G = MQ^G \). An example is given in Figure 2 where the instances of local classes (local tuples) and the corresponding instance of the global class (global tuples) are shown.

III. DATA LINEAGE

In this section we first recall the definition of Lineage-CS and PI-CS Provenance given in [2] (section III-A), then we apply these definitions to the data fusion operator MQG of MOMIS; in this way we can discuss and compare Lineage-CS and PI-CS provenance for some queries on G (section III-B).

We then introduce, in section III-C the rules to generate PI-CS Provenance for queries on G.

A. Lineage-CS and PI-CS Provenance (from [2], [16])

Lineage-CS models the provenance of a result tuple of a query \( q \) as a list \( W(q, t) = < Q_1^*, \ldots, Q_n^* > \) of subsets \( Q_i^* \) of the inputs \( Q_i \) of the query (where the inputs could be base relations or the result of other queries) that contribute to \( t \). This way of modeling provenance as independent sets of tuples presents the disadvantage that the information about which input tuples were combined to produce a result tuple is not modeled. For instance, consider a query \( q = \prod_{i} (R \Join_{a=b} S) \): an output tuple \( t \) may be produced from several outputs of the
join that are all projected on $t$. Lineage-CS would represent the provenance of $t$ as two subsets of $R$ and $S$ containing the tuples from $R$ and $S$ that were joined by the query and projected on $t$. Which tuples contributed to $t$ is apparent from this representation, but the information about which tuple from $R$ was joined with which tuple from $S$ is not modeled. To explicitly model which tuples were used together in the creation of an output tuple, authors in [2] changed the provenance representation from a list of subsets of the input relations to a set of witness lists, by introducing the concept of PI-CS (Perm Influence Contribution Semantics). 

PI-CS Provenance models the provenance with the concept of witness list $w$, which is an element from $(Q_1^* \times \ldots \times Q_n^*)$, with $Q_i^* = Q_i \cup \perp$. Thus, a witness list $w$ contains a tuple from each input of an operator or the special value $\perp$. The value $\perp$ indicates that no tuple from an input relation belongs to the witness list $w$ (and, therefore, is useful in modeling outer joins and unions which are both important in integration). Each witness list represents one combination of input relation tuples that were used together to derive a tuple. The complete formal definitions can be found in [2], [16].

B. Lineage-CS and PI-CS Provenance in MOMIS

Let $G$ be a global class and let $L(G) = \{L_1, \ldots, L_n\}$ be the set of its local classes. Given a query $q$ on $G$ we want to express the provenance for $q$ in terms of the local classes of $G$, thus:

- the Lineage-CS of $q$ is represented as a list $<L_1^*, \ldots, L_n^*>$ of subsets $L_i^*$ of the local classes $L_i$ of $G$.
- the PI-CS provenance of $q$ is represented as a set of witness lists $w$, where each $w$ contains a tuple from each $L_i$ of $G$ or the special value $\perp$ which indicates that no tuple from a local class belongs to $w$, i.e., a witness list $w$ is an element from $(L_1^* \times \ldots \times L_n^*)$, with $L_i^* = L_i \cup \perp$.

The set of local classes $\{L_1, \ldots, L_n\}$ of $G$ is not ordered; in the above definitions we can assume an arbitrary and predefined ordering of the local classes since the order of local classes in the full outer join evaluation of $MQ^G$ is not relevant.

In [2], [16] there are the formal definitions of Lineage-CS and PI-CS provenance for a generic relational operator, which can be applied to the $MQ^G$ operator too. In the following we discuss and compare Lineage-CS and PI-CS provenance for some queries on $G$. We use $L(id)$ to denote the tuple $t$ of a local class $L$ with object identifier $ID$ equal to $id$, i.e. $t[ID] = id$.

Query $q_1$ in Figure 5 is interesting since it is equivalent to the union $\pi_{ID}(L_1) \cup \pi_{ID}(L_2) \cup \pi_{ID}(L_3)$. With the Lineage-CS, the provenance of the tuple of $Q_1$ with $ID = 1$, denoted by $Q_1(1)$ would be a single list of local class tuples that contribute to deriving $Q_2(1)$. As discussed in TRAMP [2], this is a bit misleading as it indicates that all these tuples used together influence $Q_1(1)$, when in fact each, independently, influences $Q_1(1)$: PI-CS provenance captures this intuition by defining the provenance as $\{\langle L_1(1), \perp, \perp \rangle, \langle \perp, L_2(1), \perp \rangle, \langle \perp, \perp, L_3(1) \rangle\}$ (see Figure 3).

In other words, while with the Lineage-CS only a single list corresponding to the maximal subsets [1] is considered as the derivation of $Q_1(1)$, with the PI-CS provenance all the possible derivations are indicated.

Query $q_2$ of Figure 4 selects one-to-one global attributes, i.e., global attributes (such as $A$ and $B$) that are mapped only into a local class ($L_1$ and $L_2$ respectively).

With the Lineage-CS, the provenance of the first global tuple $Q_2(1)$ should be a single list of local class tuples, while with the PI-CS provenance three lists corresponding to the three possible derivations of $Q_2(1)$ are obtained. Moreover, with the Lineage-CS, the provenance for the three global tuples $Q_2(2)$, $Q_2(5)$ and $Q_2(6)$, is a single list with a local tuple coming from $L_1$ and another one from $L_2$. The PI-CS provenance provides a more precise explanation and differentiates the derivations of these three tuples: $Q_2(2)$ has a unique derivation involving both local tuples, $Q_2(5)$ has two possible derivations, each involving only one local tuple, and $Q_2(6)$ has a unique derivation involving only one local tuple.
We will now discuss and compare what Lineage-CS and PI-CS provenance produce for query with resolution functions, that represent the most significant aspect in our data fusion framework. To this end, we will consider the query $q_3$ of Figure 5 which selects the one-to-many global attribute $C$ defined by the AVG resolution function.

The simplest and most effective example to show the difference is considering the tuple $(Q_3(1))$ in Figure 5. With the Lineage-CS, the provenance of $Q_3(1)$ is a single list of local class tuples, while with the PI-CS provenance two lists with all its possible derivations are obtained.

This example confirms that the PI-CS provenance produces all the possible derivations also for a global tuple obtained by means of an AVG resolution function, with a behaviour that is homogeneous with the case of the union and one-to-one attributes discussed before. This is true in the case of the average function, but the application of the PI-CS provenance need to be further analyzed for different types of resolution functions.

In [14], the properties of the resolution functions are examined; in particular, resolution functions and subdivided into mediating and deciding functions. A function is mediating if $RF(v_1, \ldots, v_n) = y$, meaning that a new value is created by the resolution function. Intuitively, for some mediating functions, such as AVG and MEDIAN, the PI-CS provenance provides several witness lists, while for other ones, such as SUM and CONCAT, a unique witness list is produced.

Deciding functions choose among the already present values, e.g., COALESCE or SHORTEST, where $RF(v_1, \ldots, v_i, \ldots, v_n) = v_i, i \in \{1, \ldots, n\}$. As observed in [14], if we assume that ties (e.g., two shortest values) are broken by a secondary criterion, e.g., the order of the values, we always get a defined result. Intuitively, for deciding functions, the PI-CS provenance provides a unique witness list with only the local tuple whose value is chosen by the resolution function.

Taking into consideration the aspects previously described, we believe that the PI-CS provenance is more suitable in a data fusion context as it provides the users that query the integrated data with explanations for the answers they receive, but it also allows evaluating how the data obtained from a data integration system can be affected when one or more local sources become unavailable.

Therefore, in the next section we discuss the method to compute the PI-CS provenance for the $MQ^G$ operator.

C. PI-CS Provenance generation

TRAMP is implemented as an extension to the Perm relational provenance management system [10], [16] that supports PI-CS for data provenance. The method proposed in Perm to compute the PI-CS provenance is based on construction rules, that are given for each relational operator. In this section we introduce construction rules for the $MQ^G$ operator.

Let $G$ be a global class and let $C(G) = \{L_1, \ldots, L_n\}$ be the set of its local classes. Given a subset of global attributes $S = \{G_1, \ldots, G_k\}$, $MQ^G_S$ denotes $MQ^G$ with only ID $\cup$ $S$ in the select-clause. For example $MQ_{A,C}$ denotes

\[
\]
Given a global attribute $GA$ of $G$, let $R = MQ^G_{GA}$; given a value $id \in \pi_{1D}(R)$, the PI-CS provenance for the tuple $R(id) \in R$ is generated according to the following construction rules (in the following a witness list $w$ will be denoted in a simplified form, by omitting the $\perp$ elements):

- if $GA = ID$ is the object identifier:
  $$(MQ^G_{DA}, R(id)) = \{ \langle L(id) \rangle \mid L \in \mathcal{L}(G), L(id) \in L \}$$

- if $GA$ is one-to-one
  $$(MQ^G_{DA}, R(id)) = \{ \langle L(id) \rangle \mid L \in \mathcal{L}(G), L(id) \in L \text{ and } R(id)[GA] = \text{NULL} \text{ or } L_i(id)[GA] = r(id)[GA] \}$$

- if $GA$ is a one-to-many, mapped into $k \leq n$ local classes and defined by the resolution function $RF$:
  $$(MQ^G_{DA}, R(id)) = \{ \langle L(id), \ldots, L_q(id) \rangle \mid q \leq k, L_i(id) \in L_i, 1 \leq i \leq q \text{ and } RF(L_i(id)[GA], \ldots, L_k(id)[GA]) = R(id)[GA] \}$$

The first rule is trivial. The second rule take into account that, due to the full outer join operation, a NULL value for $r(id)[GA]$, can come either from a NULL value from the local class where $GA$ is mapped (i.e. $L_i(id)[GA]$) or can be obtained from any local class where $GA$ is not mapped. The last rule take into account a one-to-many global attribute $GA$; the definition of $MQ^G_{DA}$ says that $R(id)[GA] = RF(v_1, v_2, \ldots, v_n)$, with $v_i = L_i(id)[GA], 1 \leq i \leq k$. The rules generates a witness list $\langle L_i(id), \ldots, L_q(id) \rangle$ for each subset $v_1, v_2, \ldots, v_k, q \leq k$, such that $RF(v_1, v_2, \ldots, v_q)$, $RF(v_1, v_2, \ldots, v_k)$.

The construction rule for for a subset $S = \{GA_1, \ldots, GA_k\}$ with $k \geq 1$, is the following. Let $R = MQ^G_S$; for a tuple $R(id) \in R$ the PI-CS provenance is generated as follows:

1) The union $SW$ of witness lists for each global attribute $GA \in S$ is computed

$$SW = \bigcup_{GA \in S} (MQ^G_{DA}, R(id))$$

2) A witness list $w'$ is subsumed by a witness list $w$ iff $w'$ can be derived from $w$ by replacing some input tuples from $w$ with $\perp$. A witness list $w$ is excluded from $SW$ if there is a witness list $w'$ subsumed by $w$.

As an example, the PI-CS provenance of the global tuple $G(1)$ of $G = MQ^G_{I,D,A,B,C}$ can be computed as:

1) $$(MQ^G_{DA}, G(1)) = \{ \langle L_1(1), \langle L_2(1),\langle L_3(1) \rangle \rangle \rangle \}, \langle GA \in \{ID,A,B\} \rangle$$

2) $$(MQ^G_{DA}, G(1)) = \{ \langle L_3(1), \langle L_1(1),L_2(1) \rangle \rangle \}$$

$$SW = \{ \langle L_1(1),L_2(1) \rangle, \langle L_3(1) \rangle \}$$

2) $\langle L_1(1) \rangle$ and $\langle L_2(1) \rangle$ are eliminated from $SW$ since subsumed by $\langle L_1(1),L_2(1) \rangle$.

We then finally obtain $\{ \langle L_1(1),L_2(1) \rangle, \langle L_3(1) \rangle \}$.

To generate the PI-CS provenance for a generic query on $G$, we will use the above rules to calculate the provenance for the $MQ^G$ operator together with the rules in Perm [10], [16] for other operators. As an example, query $q_4$ with a WHERE condition (Figure 6) is equivalent to $\Pi_B(\sigma_{A>1}(MQ^G_{B,A}))$, we can thus use the above mentioned rules to calculate the provenance for $MQ^G_{B,A}$, while applying the rules introduced in TRAMP for selection and projection.

As an another example, queries $q_5$ (Figure 7) and $q_6$ (Figure 8) show the difference, at the level of PI-CS provenance, between bag semantics ($q_5$) and set semantics ($q_6$).

All the queries considered so far are examples of SELECT-PROJECT queries on a single global class where we showed how PI-CS provenance provides all the possible derivations for a certain tuple, which means that every witness list provides a possible derivation.

Finally, let us consider an example of SELECT-PROJECT-JOIN queries on two or more global classes. To this end, suppose we have another global class $G'$ with the same structure of $G$, i.e. $L(G') = \{L_1, \ldots, L_n\}$ where each $L_i$ has the same schema as $L_i$. Suppose then that $G'$ has the same mapping table and the same instance of $G$. In the query $q_7$ a join between $G$ and $G'$ is performed:

$$Q7 : SELELCT G.ID \\ FROM G \\ JOIN G' ON (G.C=G'.C) \\ WHERE G'.B = 7$$

The PI-CS provenance for $q_7$ is shown in Figure 9.
For definition, the P1-CS provenance is traced with respect to local classes; besides this type of provenance, for SELECT-PROJECT-JOIN queries on two or more global classes, it may be useful to trace provenance just with respect to the global classes. This corresponds to the concept of limited provenance scope, introduced in the Perm system [10], [16], to handle a from-clause item as a base relation; in particular, it can be applied if a user does not want to trace provenance down to the base relations, but is interested in the influence a view had on the query results.

We can apply the same technique in our case: for each global class \( G \) in a query, we can specify if the provenance must be traced down to the local classes of \( G \) or not (i.e. the provenance must be traced just with respect to the global class \( G \)). As an example, for the above query \( q_7 \), in the first column of figure 10 the provenance is traced with respect to global classes \( G \) and \( G' \), while in the second column the provenance is traced down to the local classes of \( G \) and with respect to the global class \( G' \).

![Fig. 10. Limited provenance scope for \( q_7 \)](image)

<table>
<thead>
<tr>
<th>ID</th>
<th>P1-CS provenance</th>
<th>P1-CS provenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{ ( (G(1),G(6)) ) }</td>
<td>{ ( (L_1(1),L_2(1),G(6));(L_3(1),G(6)) ) }</td>
</tr>
<tr>
<td>5</td>
<td>{ ( (G(5),G(6)) ) }</td>
<td>{ ( (L_1(5),G(6)) ) }</td>
</tr>
<tr>
<td>6</td>
<td>{ ( (G(6),G(6)) ) }</td>
<td>{ ( (L_1(6),L_2(6),G(6)) ) }</td>
</tr>
</tbody>
</table>

IV. CONCLUSION AND FUTURE WORK

In this paper we presented our work in progress to apply data provenance techniques in the Data Fusion framework of the MOMIS Integration System. We focused our attention on the extension of the concept of P1-CS provenance to deal with Resolution Functions.

In particular, future work will be directed in the following directions:

- **Querying Data Lineage** In a data fusion scenario, the data lineage can be useful to understand the relation between the results we obtain querying a global class \( G \) and the local classes \( G \) is mapped on. This is particularly important for example to evaluate how the data we obtain from a data integration system can be affected when one or more local sources become unavailable.

  It is thus necessary to allow querying data lineage, providing an appropriate method to express conditions in our queries to consider tuples with lineage from certain local classes.

  For example, the results from query \( q_5 \) (figure 7) show no differences between a NULL value coming from the data sources (first tuple) and a NULL value obtained because the attribute has no mapping on a local source (second tuple). Having the possibility to query the data lineage, we can have different results in these two cases.

- **Where-lineage** Our preliminary work on data lineage started with analyzing the Why-Provenance; the next step will be analyzing also the Where-Provenance, with particular regards to resolution functions. The starting point will be the observation in [14]: mediating resolution functions does not allow evaluating the Where-Lineage, while it is possible to assign it with deciding resolution functions.

- **Implementation** In the Perm relational provenance management system [10], [16] provenance and data are processed on the same data model through query rewriting. In the MOMIS system, to answer a query over a global class \( G \), the query must be rewritten as an equivalent set of queries expressed on the local schemas (local queries); this query translation performs some query optimization techniques, such as predicate push down (to push a constraint on local queries) and full join simplification (to reduce full join to left/right/inner join). Thus, our idea is to extend our query processing to include also the provenance computation in the query rewriting, in order to be able to provide the user with lineage information when obtaining the results.

REFERENCES


