Resolving Structural Conflicts in the Integration of XML Schemas: A Semantic Approach

Xia Yang Mong Li Lee Tok Wang Ling

School of Computing, National University of Singapore
3 Science Drive 2, Singapore 117543
{yangxia, leeml, lingtw}@comp.nus.edu.sg

Abstract. While the Internet has facilitated access to information sources, the task of scalable integration of these heterogeneous data sources remains a challenge. The adoption of the eXtensible Markup Language (XML) as the standard for data representation and exchange has led to an increasing number of XML data sources, both native and non-native. Recent integration work has mainly focused on developing matching techniques to find equivalent elements and attributes among the different XML sources. In this paper, we introduce a semantic approach to resolve structural conflicts in the integration of XML schemas. We employ a data model called the ORA-SS (Object-Relationship-Attribute Model for Semi-Structured Data) to capture the implicit semantics in an XML schema. We present a comprehensive algorithm to integrate XML schemas. Compared to existing methods, our algorithm adopts an n-nary integration strategy that takes into account the data semantics, importance of a source, and how the majority of the sources model their data when resolving structural conflicts such as attribute/object class conflict and ancestor-descendant conflict. Further, redundant object classes and transitive relationship sets are removed to obtain a more concise integrated schema.

1 Introduction

Advances in the Internet infrastructure have facilitated access to large amounts of information sources. Many of these sources are heterogeneous, and an integrated access to these sources remains the focus of ongoing research. Much work has been done on the integration of relational databases, ranging from semantic enrichment using a semantic data model such as the Entity-Relationship model or the object-oriented data model, translation algorithms, and conflict resolution [8][9][10][22]. Integration systems such as [4][7][15][16][18][21] have also been developed.

The adoption of the eXtensible Markup Language (XML) as the standard for data representation and exchange has led to an increasing number of XML data sources, both native and non-native. Native XML data sources are essentially XML files with an associated XML schema, while non-native XML sources such as the relational database publish their data in XML format together with the XML schema. Given the semistructured nature of XML data that can be modeled as a tree or a graph, recent research in integrating XML data sources has mainly concentrated on schema matching [4][12][21].

The task of integrating XML data sources is non-trivial for the following reasons:

1. The XML Schema or DTD is lacking in semantics. While this has prompted proposals to augment the schema with information such as keys [3], and functional dependencies [11], it remains unclear whether the relationship between the element objects is binary or n-nary, and whether an attribute belongs to an element object class (e.g. title of an element book) or to the relationship set between elements (e.g. quantity of books supplied by a supplier to a bookshop).
2. The source schemas are heterogeneous, containing various conflicts involving naming conflict, cardinality conflict, and structural conflict such as attribute/object class conflict and ancestor-descendant conflict. There is no unique global schema, but it is subject to the needs of applications and the perspective of the users.

To address these issues, we develop a semantic approach to the integration of XML schemas. We employ the semantically rich model ORA-SS [5] for semistructured data to capture the semantics of the underlying XML data. An n-nary integration strategy that provides a global view of the source schemas is adopted. The integrated schema obtained takes into consideration underlying data semantics such as different relationship sets among equivalent object classes, the importance of the source schemas, and how the majority of the sources schemas modeled their data. Structural conflicts such as attribute-object class conflict and ancestor-descendant conflict are resolved in the process. Finally, redundant object classes and transitive relationship sets are identified and removed to obtain a more concise integrated schema.

The rest of the paper is organized as follows. Section 2 presents some background material including a brief description of the ORA-SS model. Section 3 gives a motivating example and highlights the various features that we consider in our integration strategy. Section 4 describes the details of the algorithm to integrate XML schemas. Section 5 discusses related work, and we conclude in Section 6.

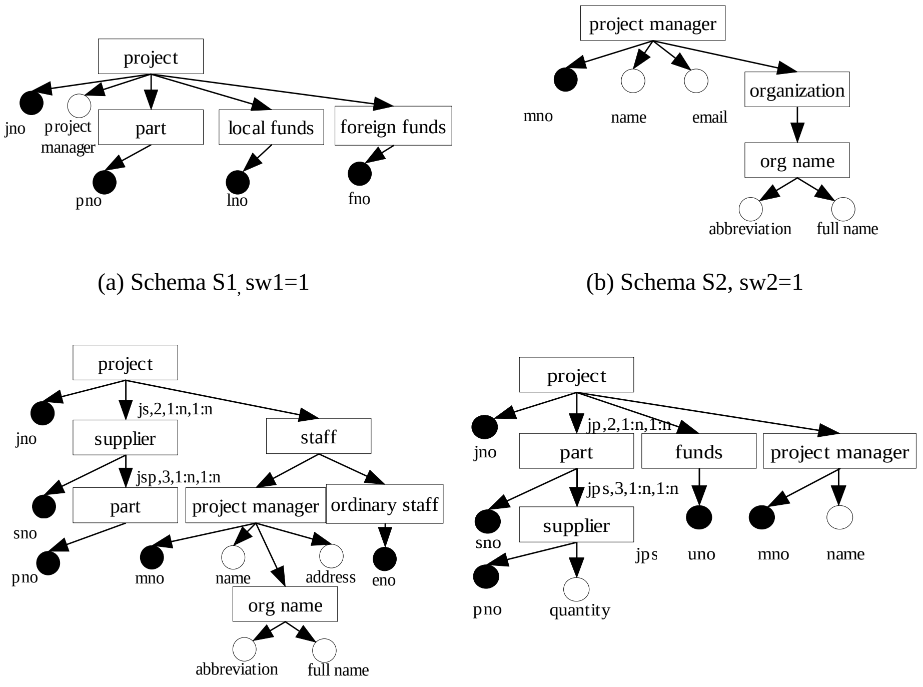
2 Preliminaries

In this section, we first describe the ORA-SS model that we utilize in our integration strategy. This is followed by the assumptions we make in our integration approach.

2.1 ORA-SS Model

The ORA-SS model (Object-Relationship-Attribute model for Semi-Structured data) is a semantically rich data model that has been designed for semi-structured data [5]. The rich semantics of ORA-SS allows us to capture more of the real world semantics, and use them for integration. The ORA-SS model distinguishes between objects, relationships and attributes. The relationships between objects are expressed explicitly. An object class in the ORA-SS model is similar to the concept of entity type in an ER model and classes in the object-oriented model. They coincide with the concept of elements in XML. An object class may be related to another object class through a relationship set. Attributes are properties, and may belong to an object class or a relationship set.

Here, we use the ORA-SS Schema Diagram as the conceptual model for XML data. The object classes such as "project" and "part" in Fig 1(d) are represented by labeled rectangle. The relationship set between the object classes are denoted by *name, n, p, c*, where *name* denotes the name of the relationship, *n* is the degree of the relationship, *p* is the participation constraint of the parent object class in the relationship, and *c* is participation constraint of the child object. The participation constraints are defined using the min:max notation. The labeled circles denote attributes, and the filled circles denote keys. Attributes are properties of object class or the relationship set. For example, in Fig 1(d), "jno" is the attribute of object class "project", while "quantity" is the attribute of relationship set "jps". The degree of relationship set "jps" is 3, which is a ternary relationship set involving object classes "project", "part" and "supplier". For details on ORA-SS, please refer to [5].



(a) Schema S1, sw1=1

(b) Schema S2, sw2=1

(c) Schema S3, sw3=7

(d) Schema S4, sw4=1

Fig. 1. ORA-SS Schema Diagrams for four XML sources.

2.2 Assumptions

The input to the proposed integration algorithm is a set of ORA-SS schemas, which has been generated from XML schemas. Details of the transformation of XML schema to the ORA-SS model are given in [2]. Inputs from the users may be solicited to enrich the ORA-SS schema with the necessary semantics.

The output of the algorithm is an integrated schema, also modeled in ORA-SS. Since queries on the integrated schema will be subsequently mapped to equivalent queries on the data sources, the integrated schema should contain all the information modeled in the original schemas. Further, the integrated schema should be as simple and concise as possible to facilitate users' understanding.

For meaningful integration to occur, we assume that the various sources model similar domains. Object classes with the same label are considered to be semantically equivalent, that is, they refer to the same object class in the real world. Similarly, attributes of the same object class (or relationship set) with the same label are also semantically equivalent, that is, they refer to the same property of an object class (or relationship sets) in the real world. The object classes (or relationship sets) in the different original schemas that refer to the same real world object (or relationship) may have different names. We assume that the renaming step have been done before the integration process. Note that there may also exist different relationship sets between the same object classes. In such cases, we assume they will be assigned different labels.

3 Motivating Example

In this section, we illustrate some of the unique features of the integration strategy we offer. Consider the ORA-SS schema diagrams for four XML sources in Fig 1. The swi under each schema indicates the source weight, i.e., the importance of a source. This is determined by users or computed based on some statistic information.

A. Resolve attribute-object class conflict.
This occurs when a concept has been modeled as an attribute in one schema, and as an object class in another schema. For example, the attribute "project manager" in schema S1 is semantically equivalent to the object class "project manager" in schema S2 of Fig 1. This conflict can be easily resolved by mapping the attribute to an object class (see Fig 2).

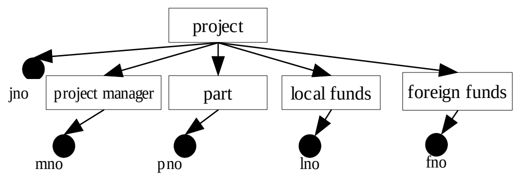


Fig. 2. Schema S1': Attribute "project manager" in schema S1 of Fig 1 has been transformed into an object class "project manager" in S1'.

B. Resolve generalizations and specializations.
A generalization exists when an object class in one schema is the union of several object classes in another schema. Consider again Fig 1. The object class "funds" in schema S4 is a generalization of the object classes "local funds" and "foreign funds" in schema S1. The integrated schema will include the generalization hierarchy as shown in Fig 3.

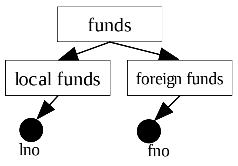


Fig.3. Build a generalization hierarchy from S1 of Fig 1.

C. Merge the schemas to obtain an integrated graph.

Fig 4 shows the graph obtained from merging the schemas S1', S2, S3 and S4. Each node in the graph denotes an object class, and edges represent the relationship sets between the object classes. To facilitate processing, attributes are first omitted from the integrated graph. The attributes will be incorporated into the final integrated schema. Note that only the equivalent relationship sets will merged together. Semantically different relationship sets between the equivalent object classes will be treated as different relationship sets, as indicated by the different edges.

The edges in the integrated graph are weighted as follows. Since we have "project" as the parent of "part" in schemas S1 and S4, the weight of the edge from "project" to "part" is given by the sum of the weights of these schemas, that is, 1+1=2. In the same way, since "project" is the parent of "staff" in schema S3 only, the weight of this edge is 7. Since the edge from "supplier" to "part" in S3 is actually involved in two relationship sets jsp and sp, its edge weight would be given by 7*2=14.

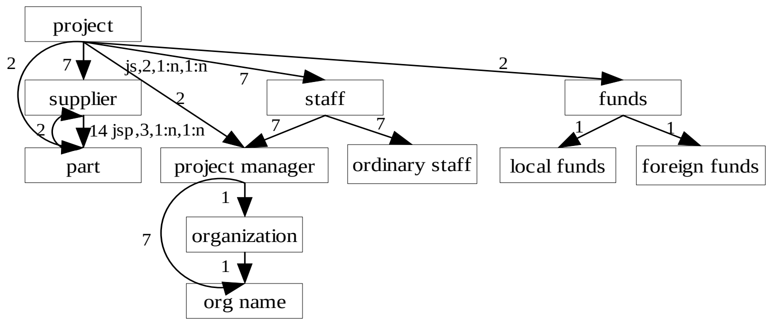


Fig. 4. Integrated graph obtained from the schemas in Fig 1.

D. Transform integrated graph to resolve structural conflicts and remove redundancy.

We proceed to transform the graph to differentiate the semantically different relationships between equivalent object classes, identify cycles to resolve ancestor-descendant conflicts, remove redundant object classes and redundant relationship sets. Redundant relationship sets include relationship sets that are derived from projecting higher-degree relationships in the schema and transitive relationship sets.

D-1. Differentiate semantically different relationship sets between equivalent object classes.

Consider the schemas S5 and S6 in Fig 5 that are structurally the same, except for the additional object class "contract" in S6. The relationship sets between the same object classes are semantically different. The relationship set in schema S5 indicates that the person owns the house, while that in schema S6 indicates that the person rents the house. We first merge the two schemas to obtain the integrated graph G56 before transforming it to G56' (see Fig.5). The edges from object classes "house1" and "house2" to the object class "house" in G56' indicate foreign key-key references. Note that the relationship phc between the "person", "house" and "contract" is represented explicitly in the transformed graph.

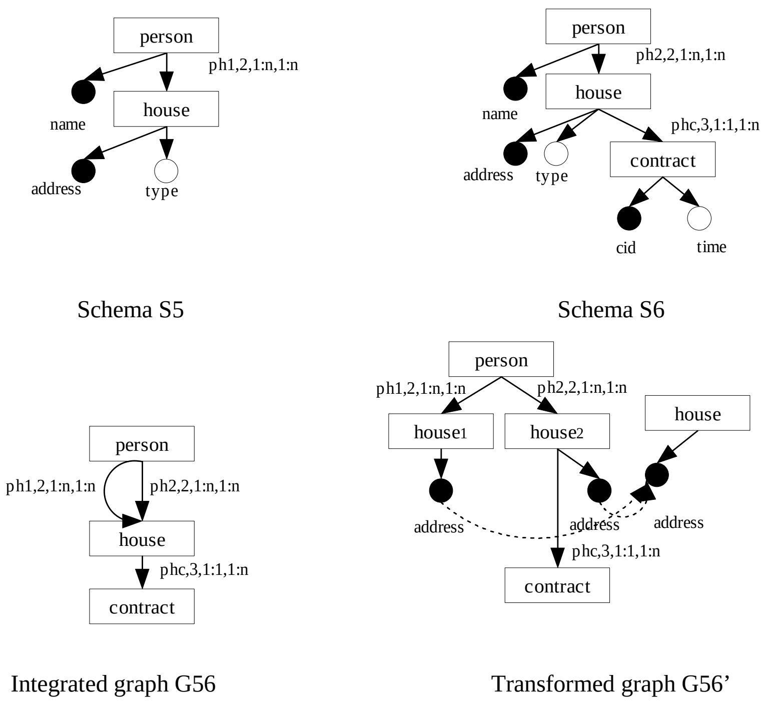


Fig. 5. Different relationship sets among equivalent object classes.

D-2. Remove relationship sets that are projections of higher degree relationship sets.

A schema may model a relationship set that is a projection of another relationship set in another schema. For instance, if we integrate the schemas S1 and S3, the integrated graph will contain the binary relationship set between "project" and "part" from schema S1, and the ternary relationship set between "project", "supplier" and "part" from schema S3. Since the former is a projection of latter relationship set, we remove the binary relationship set and keep the ternary relationship set in the integrated graph. Subsequently, we can issue a query "/project//part" on the integrated schema to retrieve all the "part" information.

D-3. Resolve ancestor-descendant conflicts.

An ancestor-descendant conflict arises when a schema models an object class A as an ancestor of another object class B, and the other schema models B as the ancestor of A. The simplest form of this conflict is the parent-child conflict in schemas S3 and

S4. We have “supplier” as the parent of “part” in S3, while “part” is the parent of “supplier” in S4. This conflict creates a cycle “supplier” → “part” → “supplier” in the integrated graph of Fig 4. One of the edges which represent the inverse relationship sets can be removed to break the cycle. We propose to remove the edge with the lowest edge weight, that is, the edge from the less important schema. In this case, the edge from “part” to “supplier” with an edge weight of 2 will be removed.

Fig 6 shows another example of an ancestor-descendant conflict. The object class “module” is the ancestor of “tutor” in schema S7, while “tutor” is the ancestor of “module” in S8. This conflict will create a cycle in the integrated graph G78. The conflict can be resolved by removing one of the edges that has the least weight. Further, the edge removed should represent a relationship set that can be derived by a series of joins and projections of the other relationship sets involved in the cycle.

If the source weights are sw7=2, sw8=1, then the weight of the edge from “tutor” to “module” is 1. Since this edge has the lowest edge weight, we will remove it from G78. The transformed graph obtained at this point will be G78’.

On the other hand, if the source weights are sw7=1, sw8=2, then the weight of the edge from “tutor” to “module” is 2, and will not be removed. The weights of the edges from “module” to “lecturer”, and from “lecturer” to “tutor” are both 1. Since both of these edges have the lowest edge weight, we can remove either one of them, which will result in the transformed graph G78(a) or G78(b).

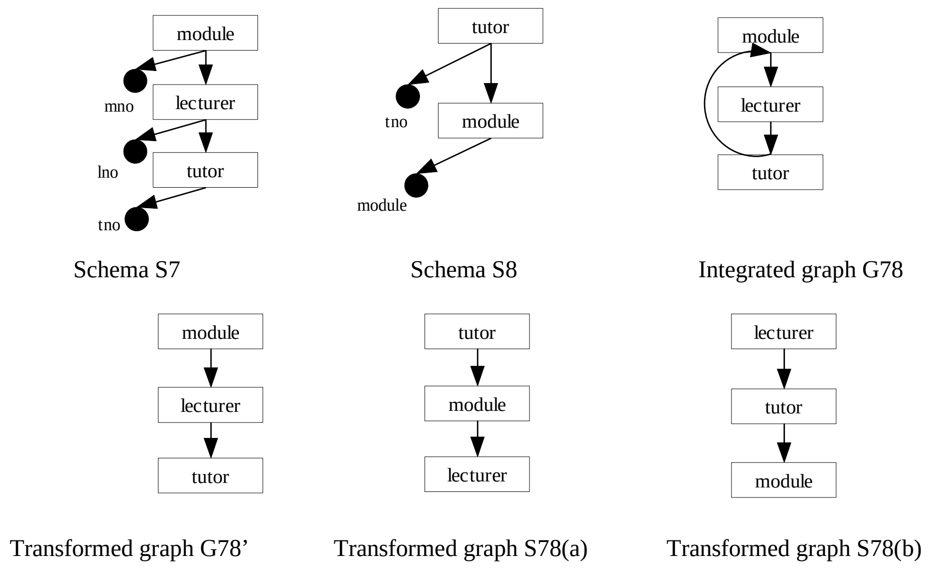


Fig. 6. Example of an ancestor-descendant conflict.

D-4. Remove transitive relationship sets.

Transitive relationships sets are also redundant, and can be removed so that the resulting integrated graph will be concise. For example, the relationship set between “project” and “project manager” in Fig 4 is a transitive relationship set that can be obtained from the relationship sets between “project” and “staff”, and between “staff” and “project manager”. Thus, we can remove the transitive relationship set from the integrated graph.

Fig. 4 also contains another transitive relationship set between “project manager” and “org name”. We observe that the object class “organization” does not have any attribute, and has only one child object class “org name”. This object class from schema S2 cannot contain any instances in the corresponding XML data files. Since “organization” is a redundant object class, we propose to remove it and its associated relationship sets from the integrated graph in Fig 4. As a result, the relationship set between “project manager” and “org name” is no longer a transitive relationship set.

D-5. Remove multiple parent nodes.

If a node has more than one incoming edges in an integrated graph, then it is called a *multiple parent* node. Consider the integrated graph G9-10 in Fig 7. The two incoming edges to “student” indicate two different relationship sets. The attribute “mark” can only belong to one of them, namely, the relationship set “jd”. In the transformed graph G9-10’, we will split the multiple parent node and represent these two relationship sets separately.

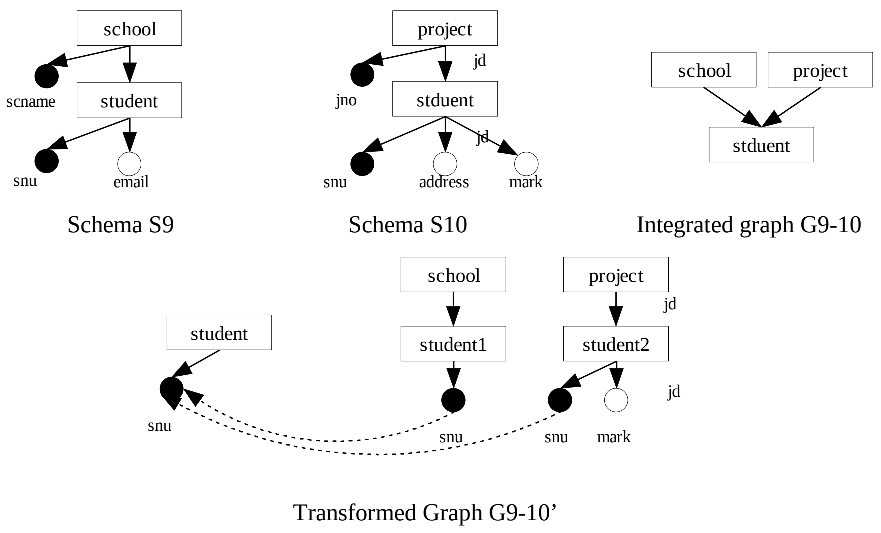


Fig. 7. Example of a multiple parent node.

Fig 8 shows the transformed graph obtained for the source schemas in Fig 1 after addressing the above concerns. For instance, when solving ancestor-descendant conflict, the cycle “supplier”→“part”→“supplier” is detected and the edge “part”→“supplier” is deleted. The redundant object class “organization” and its associated edges are deleted. Transitive edges as “project”→“project manager” and “project”→“part” are also removed. The transformed graph is augmented with attributes such as “quantity” for the ternary relationship set “jsp”. The final integrated schema is shown in Fig 9. Note that the attribute “quantity” belongs to the relationship set “jps” in schema S4 (see Fig. 1), which is a ternary relationship set associating object classes “project”, “ supplier” and “part”. Since the node “part” is at the lowest level compared to "supplier" and "project", the attribute "quantity" becomes an attribute under "part".

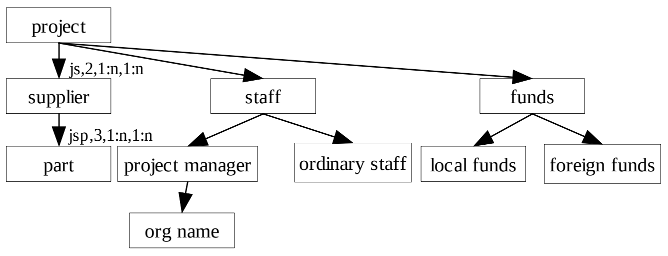


Fig. 8. Transformed graph obtained from Fig 4.

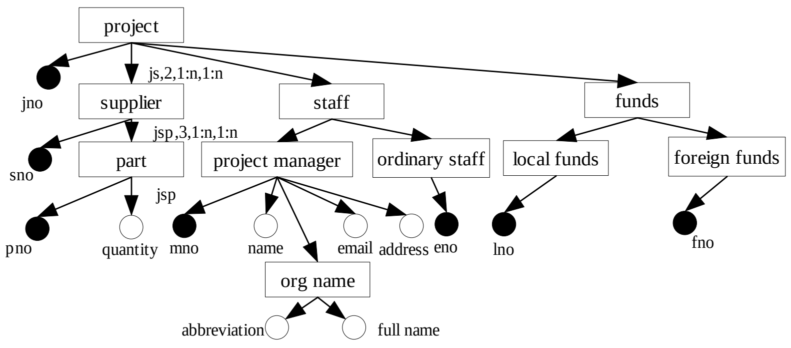


Fig. 9. Final integrated schema.

4. Integration Algorithm

In this section, we first discuss some of the terms used before giving the details of the integration algorithm.

If a node i has more than one incoming edges in an integrated graph, then we called it a *multiple parent* node. If a directed edge sequence <$e_{i0,i1}$, $e_{i1,i2}$, …… $e_{im,i(m+1)}$, $e_{i(m+1),i0}$> occurs in an integrated graph, then a *cycle* exists. We observe that an ancestor-descendant conflict occurs if and only if there is a cycle in the integrated graph.

There are essentially four main steps in our integration algorithm:

1. Preprocessing.
2. Construct integrated graph.
3. Transform graph.
4. Augment graph with attributes.

The input is a set of schemas modeled using the ORA-SS model. The output is an integrated ORA-SS schema. The third step *Transform Graph* aims to identify semantically different relationships among equivalent object classes, resolve ancestor-descendant conflicts, and remove redundant object classes and redundant relationship sets such as transitive relationship sets. The resulting integrated schema preserves data semantics in the sources, considers how the majority of the sources model the data, and is concise.

Step 1 Preprocessing.

1.1 Resolve attribute-object class conflict.
If the same concept is expressed as an object class in one schema, and as an attribute in another schema, then convert the attribute to an equivalent object class. The attribute becomes the key of this new object class.

1.2 Resolve generalizations and specializations.
When one object class is the generalization object class of some object classes of other schemas, it becomes the parent node of these object classes.

Step 2 Construct Integrated Graph

2.1 Merge the equivalent object classes and relationship sets from original schemas to obtain an integrated graph G such that each node is an object class, and edges denote relationship sets between the object classes. Note that attributes are not included in G.

2.2 Compute the weights of the edges.

```
For each edge e in G do
    Let $e_1$, $e_2$,… $e_k$ be the equivalent edges in the original schemas $s_1$, $s_2$, …$s_k$.
    Let $sw_1$, $sw_2$, … $sw_k$ be the source weights of the schemas $s_1$, $s_2$, …$s_k$ respectively.
    Let $n_1$, $n_2$, … $n_k$ be the number of relationship sets the edge is involved in the schemas $s_1$, $s_2$, …$s_k$
    Set the weight of the edge $ew = sw_1*n_1+sw_2*n_2+ … sw_k*n_k$.
```

Step 3 Transform Graph

3.1 Differentiate semantically different relationship sets between equivalent object classes.

```
For each node $n_s$ in G do
    If $n_s$ has k outgoing edges {$e_{s1}$, $e_{s2}$, …, $e_{sk}$} to the same node $n_t$ Then
        Create k duplicate nodes {$n_{t1}$, …, $n_{tk}$} of $n_t$;
        Each edge $e_{si}$ (from $n_s$ to $n_t$), $1 \le i \le k$, becomes an edge from $n_s$ to $n_{ti;}$
        For each $n_{ti}$, $1 \le i \le k$, do
            Create a foreign key-key reference from the key of $n_{ti}$ to that of $n_t$.
        For each child node c of node $n_t$ do
            If c is involved in an n-nary relationship set that includes $e_{si}$
            Then   Move c and its descendent nodes from $n_t$ to $n_{ti}$ .
```

3.2 Remove relationship sets that are projections of higher degree relationship sets.

```
    For each n-nary relationship set R in G do
    Let N = {$n_1$, …, $n_k$}  be the set of nodes involved in relationship R.
    For each relationship set R’ that involves a subset of nodes in N do
        If          R’ is a projection of R
```

```
            Then    Remove R' from the integrated graph.
3.3 Resolve any ancestor-descendant conflicts which create cycles in G.
    For each cycle in G do
        Let $e_{ij}$ be the edge with the smallest edge weight in the cycle.
        If $e_{ij}$ can be derived from other relationship sets in the cycle.
        Then  Remove $e_{ij}$ from G.

3.4 Remove redundant relationship sets and redundant object classes.
    For each multiple parent node n in G do
        Let P be the set of parent nodes of n.
        While $|P| > 1$ do
            Let $p_{max} \in P$
            Let $<n_0, n_1, \ldots, n_k>$ be the path from $p_{max}$ to n, where $n_0 = p_{max}$, $n_k = n$,
            and $k > 1$.
            /* remove redundant object classes with no attribute and only one
            child object class. */
            For each node $n_i$ in the path, $0 < i < k$, do
                If $n_i$ has no attributes and no sub-object classes besides $n_{i+1}$
                Then  Remove $n_i$ and its associated edges from G;
                      Create an edge between $n_{i-1}$ and $n_{i+1}$;
            $P = P - \{p_{max}\}$;
            If the edge from $p_{max}$ to n can be derived from $<n_0, n_1, \ldots, n_k>$
            Then  Remove the transitive edge from $p_{max}$ to n in G.

3.5 Remove multiple parent nodes.
    For each multiple parent node $n_m$ in G do
        Let $n_m$ have k incoming edges $e_1, e_2, \ldots, e_k$ from nodes $n_1, n_2, \ldots, n_k$
        respectively.
        Create k duplicate nodes $\{n_{m1}, \ldots, n_{mk}\}$ of $n_m$;
        Each edge $e_i$ (from $n_i$ to $n_m$), $1 \le i \le k$, becomes an edge from $n_i$ to $n_{mi}$;
        For each node $n_{mi}$, $1 \le i \le k$, do
            Create a foreign key-key reference from the key of $n_{mi}$ to that of $n_m$.
        For each child node c of node $n_m$ do
            If c is involved in an n-nary relationship set that includes $e_i$
            Then    Move c and its descendent nodes from $n_m$ to $n_i$ .
```

Step 4 Augment Graph with Attributes

```
4.1 Map the transformed graph G to an equivalent ORA-SS schema S.

4.2 Augment the schema with the attributes of object classes.

4.3 Augment the schema with attributes of relationship sets.
```

5. Related Work

Research in data integration has focused on various aspects to integrate information from multiple sources. Most of the work has focused on the matching problem to find equivalent elements among the different sources. These work include XClust [12], CUPID [14], SKAT [16][17], and Xyleme [19]. A taxonomy and a survey of matching approaches are given in [6].

Having obtained a set of equivalent elements, the next step is to obtain an integrated schema. [7] use schema learning to generate a set of tree grammar rules from the DTDs in a class and optimizes the rules to transforms them into an integrated view. Fig 10 shows the integrated schema that [7] will obtain. Since the method does not take into account the underlying semantics of the data, the attribute “quantity” is considered to belong to “supplier”. Further, the relationship set between “project” to “project manager” is transitive relationship set, which is redundant. The relationship set from “part” to “supplier” and “project” to “part” is redundant. In contrast, the integrated schema obtained by our approach preserves the underlying data semantics and is concise (see Fig 9).

LSD [4] employs instance information and machine learning techniques in their integration work. This is because instances contain more information than the schemas. For example, if the phone number of a given element have significant commonalities, the phone numbers are more likely to be the office phones of employees, rather than home phones. However, the number of instances is very much larger than that of the schemas, hence this method is very costly.

All these work do not take into consideration the importance of the individual data sources, and how the majority of the local schemas model their data. In contrast, our proposed method employs the ORA-SS conceptual model which is able to capture the semantics necessary for the resolution of structural conflict during integration. The n-nary strategy that we adopted provides a global view of the local sources, and is faster compared to the binary strategy, whose intermediate schemas will grow with the number of sources. The binary strategy will not be able to utilize the source importance and how the majority of the sources model the data.

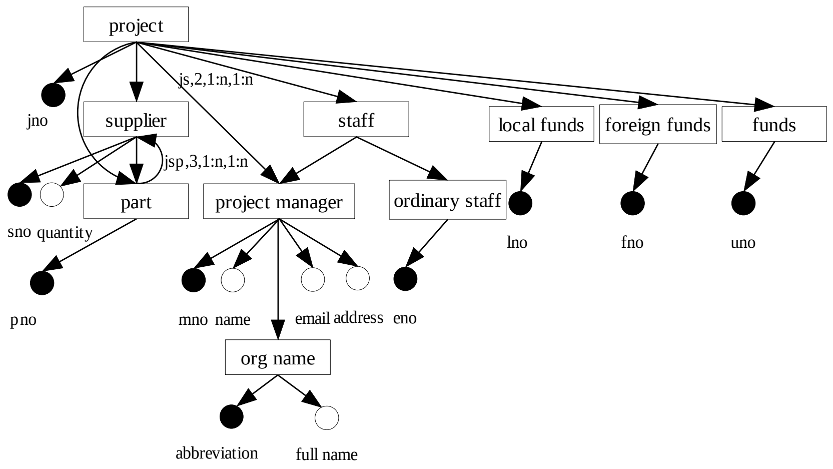


Fig. 10. Integrated schema obtained by [7].

6. Conclusion

In this paper, we have introduced a semantic approach to resolve structural conflicts in the integration of XML schemas. We employed the ORA-SS semantic data model to capture the implicit semantics in an XML schema. We presented a comprehensive n-nary algorithm to integrate XML schemas. Compared to existing methods, our algorithm takes into account the data semantics, the importance of a source, and how the majority of the sources model their data. Structural conflicts such as attribute/object class conflict, ancestor-descendant conflict are resolved in our approach. We also remove redundant object classes and relationship sets such as transitive relationship sets, and relationship sets, which are projections of higher degree relationship sets in order to obtain a concise integrated schema.

References

1. S.Castano, V. Antonellis, S. C. Vimercati, M. Melchiori. An XML-Based Framework for Information Integration over the Web. IIWAS, 2000.
2. Y.B. Chen, T.W. Ling, M.L. Lee. Designing Valid XML Views. ER, 2002.
3. P. Buneman, S. Davidson, W. Fan, C. Hara, W.C. Tan. Keys for XML. WWW, 2001.
4. A. Doan, P. Domingos, A. Levy. Learning Source Descriptions for Data Integration. WebDB, 2000.
5. G. Dobbie, X. Wu, T.W. Ling, M.L. Lee. ORA-SS: An Object-Relationship-Attribute Model for Semi-structured Data. Technical Report TR21/00, National University of Singapore, 2000.
6. E. Rahm, P. Bernstein. On Matching Schemas Automatically. MSR Tech. Report MSR-TR-2001-17, 2001.
7. E. Jeong, C.-N. Hsu. Induction of Integrated View for XML Data with Heterogeneous DTDs. ACM CIKM, 2001.
8. T.W. Ling, M.L. Lee. Relational to Entity-Relationship Schema Translation Using Semantic and Inclusion Dependencies, in Journal of Integrated Computer-Aided Engineering, John-Wiley Publishers, Vol 2, No 2, pages 125-145, 1995.
9. M.L. Lee, T.W. Ling. Resolving Structural Conflicts in the Integration of Entity-Relationship Schemas. OOER, 1995.
10. M.L. Lee, T.W. Ling. Resolving Constraint Conflicts in the Integration of Entity-Relationship Schemas. ER, 1997.
11. M.L. Lee, T.W. Ling, W.L. Low. Designing Functional Dependencies for XML, EDBT, 2002.
12. M.L. Lee, L.H. Yang, W. Hsu, X. Yang. XClust: Clustering XML Schemas for Effective Integration, ACM CIKM, 2002.
13. D. Maier. Theory of Relational Databases. Computer Science Press, 1983.
14. J. Madhavan, P.A. Bernstein, E. Rahm. Generic Schema Matching with Cupid. VLDB, 2001.
15. R. Mello, S. Castano, C.A. Heuser. A Method for the Unification of XML. Information and Software Technology Journal, 2002.
16. P. Mitra, G. Wiederhold and J. Jannink. Semi-automatic Integration of Knowledge Sources. Fusion, 1999.
17. P. Mitra, G. Wiederhold, M. Kersten. A Graph-Oriented Model for Articulation of Ontology Interdependencies. EDBT 2000.

18. F. Naumann, U. Leser, J.C. Freytag. Quality-driven Integration of Heterogeneous Information Systems. VLDB, 1999.
19. C. Reynaud, J.-P. Sirot, D. Vodislav. Semantic Integration of XML Heterogeneous Data Sources. IDEAS, 2001.
20. http://www.cogsci.princeton.edu/~wn
21. Xyleme. A dynamic warehouse for XML Data of the Web. IEEE Data Engineering Bulletin 24(2):40-47, 2001.
22. L.L. Yan, T.W. Ling. Translating Relational Schema with Constraints into OODB Schema. IFIP DS-5 Semantics of Interoperable Database Systems. 1992.