APPLICATIONS OF ORA-SS: AN OBJECT-RELATIONSHIP-ATTRIBUTE MODEL FOR SEMISTRUCTURED DATA

Tok Wang Ling[1] Mong Li Lee[1] Gillian Dobbie [2]

Abstract
Semistructured data is becoming increasingly important with the introduction of XML, related languages, and technologies. The recent shift from DTDs (document type definitions), a simple schema definition language, to XML-Schema, a more expressive schema definition language, for XML data highlights the importance of schema definitions for semistructured data applications. At the same time, there is a move to extend semistructured data models to express richer semantics. In this paper we highlight the important concepts in ORA-SS (Object-Relationship-Attribute model for Semistructured data), a semantically rich data model for semistructured data, and demonstrate how the ORA-SS data model can be used to design efficient and non-redundant storage organizations for semistructured data, and define meaningful semistructured views. ORA-SS not only reflects the nested structure of semistructured data, but also distinguishes between object classes, relationship types and attributes. It is possible to specify the participation constraints of object classes in relationship types and distinguish between attributes of relationship types and attributes of object classes. Such information is lacking in existing semistructured data models.

1 Introduction

Semistructured data is becoming increasingly important with the introduction of XML, related languages, and technologies. Usually part of the data has structure and part has no consistent structure. Initially it was claimed that semistructured data was self-describing, and hence it was not necessary to define a schema for the data. However, the recent move from DTD [1] (Document Type Definition), a simple schema definition language, to XML-Schema [19], a more expressive schema definition language highlights the importance of a schema definition for semistructured data applications. Similarly, there is a move to extend semistructured data models to express richer semantics. The data models that have been proposed specifically for semistructured data, e.g. those described in [2, 4, 7, 15], are defined for at least one of the two following purposes: for finding a common schema for two or more heterogeneous data sources or for extracting a schema from a semistructured document. There is a need for a data model with richer semantics for the purpose of recognizing redundancy in semistructured data, designing data stores for semistructured data, and defining views on semistructured data. A semantically rich data model will enable one to capture more of the real world semantics.

Semistructured data is typically stored in flat files or in repositories, such as relational database repositories [16], object-oriented repositories [13] or in specialized semistructured systems such as Strudel [6], Lore [14], Lotus Notes [12] and Tamino [17]. Like in traditional database systems,

[1]Department of Computer Science, National University of Singapore, Singapore
[2]Department of Computer Science, University of Auckland, New Zealand

efficient storage and access, and minimizing update anomalies is very important, and this cannot be done without knowledge of the semantics of the data.

In this paper we briefly describe ORA-SS (Object-Relationship-Attribute model for Semistructured data) [DoWu00], which is a semantically rich data model for semistructured data. We demonstrate how ORA-SS can facilitate the following tasks:

- Identify redundancy in semistructured data via the normalization of ORA-SS;
- Design efficient and consistent storage for semistructured data by developing algorithms that map the logical ORA-SS model to an object-relational data store;
- Describe what objects, relationships and attributes can be reached for a particular object class through a reference;
- Evaluate XML queries on a relational or object-relational database;
- Publish XML views of an underlying ORA-SS database.

The main advantages of ORA-SS over existing data models is its ability to express the degree of an n-ary relationship type, and distinguish between attributes of object classes and attributes of relationship types. Knowing the degree of an n-ary relationship type leads to more efficient storage and access to the data. Distinguishing between object class attributes and relationship type attributes enables us to describe which object class attributes and relationship type attributes are reachable from an object class that references another object class. Currently, an attribute of a relationship type is treated no differently than an attribute of an object class, and it is impossible to distinguish which attributes are reachable from a referencing object class, and which are not.

The rest of the paper is organized as follows. Section 2 briefly describes the ORA-SS data model. Section 3 gives examples to demonstrate how ORA-SS is used. Section 4 concludes the paper, highlighting some future directions of research.

2 The ORA-SS Data Model

The ORA-SS data model has three basic concepts: object classes, relationship types and attributes. An object class is similar to a set of entities in the real world, an entity type in an ER diagram, a class in an object-oriented diagram or an element in the semistructured data model. A relationship type in the ORA-SS data model represents a nesting relationship. Attributes are properties, and may belong to an object class or a relationship type. The ORA-SS data model consists of four diagrams: the schema diagram, the instance diagram, the functional dependency diagram and the inheritance diagram. The instance diagram shows the instances of object classes, relationship types and attributes. The functional dependency diagram captures the functional dependencies among object classes in relationships. The inheritance diagram depicts the class hierarchies of object classes in the schema. A fuller description of the data model can be found in [5]. In this paper, we will focus on the schema diagram.

Figure 1 shows the ORA-SS schema diagram of an instance represented in Figure 2. An object class is represented as a labeled rectangle. A relationship type between two object classes in an ORA-SS schema diagram can be described by *name*, *n*, *p*, *c*, where *name* denotes the name

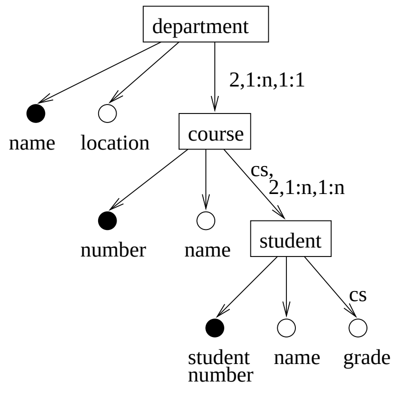

Figure 1: **ORA-SS Schema Diagram**

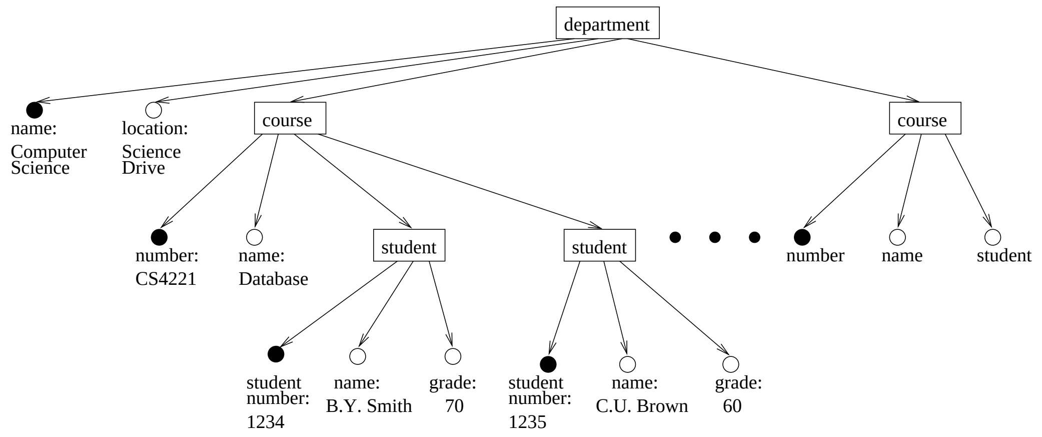

Figure 2: **ORA-SS Instance Diagram**

of the relationship type, n is an integer indicating the degree of the relationship type ($n = 2$ indicates binary, $n = 3$ indicates ternary, etc.), p is the participation constraint of the parent object class in the relationship type, and c is the participation constraint of the child object class. The participation constraints are defined using the min:max notation. Hence, 0:1, 0:n, 1:n represents ?, *, + respectively.

Attributes are denoted by labeled circles. Keys are filled circles. A special attribute name `ANY` indicates an attribute of unknown or heterogeneous structure. In ORA-SS, it is possible to distinguish between primary and candidate keys, and indicate whether a key is composite or derived. An attribute can be single-valued or multivalued. A multivalued attribute is represented using an * or + inside the attribute circle. Attributes of an object class can be distinguished from attributes of a relationship type. The former has no label on its incoming edge while the latter has the name of the relationship type to which it belongs on its incoming edge. Note that an instance of that object class or relationship type would have a subset of the attributes shown.

An object class can reference another object class via a dashed edge. Such references are useful in modeling recursive and symmetric relationships as shown in Figures 3 and 4 respectively.

One of the features of semistructured data is that some kind of ordering is enforced. For example, in XML, ordering is enforced on elements. We distinguish between 3 kinds of ordering in the ORA-SS data model:

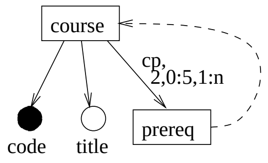

Figure 3: **Using References to Model Recursive Relationships in an ORA-SS Schema Diagram**

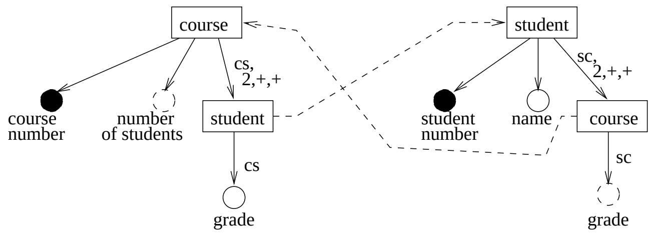

Figure 4: **Using References to Model Symmetric Relationships in an ORA-SS Schema Diagram**

- the ordering on instances of an object class,
- the ordering on values of an attribute, and
- the ordering on attributes of an object class.

A characteristic of semistructured data is that attributes and object classes are likely to be less homogeneous than in structured data. ORA-SS provides for two different kinds of disjunction:

- disjunctive object classes, and
- disjunctive attributes.

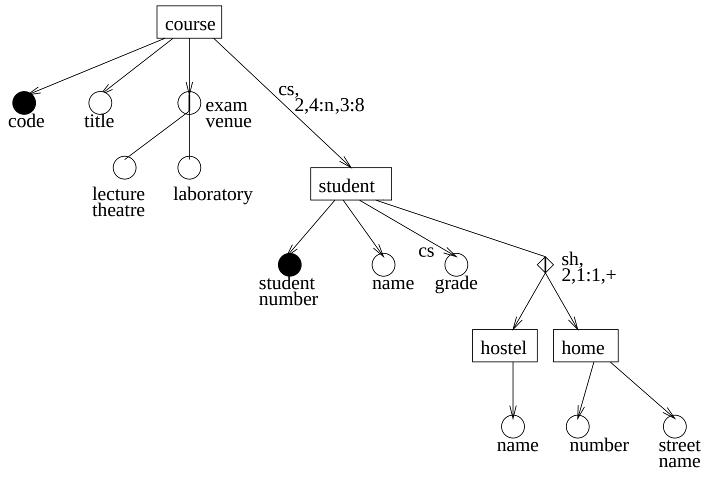

Figure 5: **Disjunctive Attribute and Relationship in an ORA-SS Schema Diagram**

A disjunctive relationship type is used to represent disjunctive object classes, such as a student lives in a hostel OR at home, and is represented by a relationship type diamond labeled with symbol “|”. A disjunctive attribute is represented by a circle labeled with symbol “|”, with edges from this circle to the alternatives. Figure 5 shows an example ORA-SS schema diagram containing disjunctive attribute and relationship.

3 Applications of ORA-SS

The richer semantics of the ORA-SS data model are important, and even necessary. In this section, we outline how the ORA-SS data model can help identify redundancy in semistructured data, define how semistructured data can be stored in a database and queried, describe how data in a relational database can be represented as semistructured data and distinguish what attributes can be reached for a particular object class through a reference.

3.1 Normal Form in ORA-SS Schema Diagram

As in traditional databases, redundancy leads to possible update anomalies in semistructured data. For example, consider the XML in Figure 6. The details of a course are repeated for each professor that teaches the course. Until a normalization theory is defined for semistructured data, the best way to identify and eliminate redundancy is to use heuristics.

```
<department>
    <professor>
        <staffnumber>12</staffnumber>
        <name>Smith</name>
        <course>
            <coursecode>230</coursecode>
            <title>Software Engineering</title>
        </course>
    </professor>
    <professor>
        <staffnumber>22</staffnumber>
        <name>Jones</name>
        <course>
            <coursecode>230</coursecode>
            <title>Software Engineering</title>
        </course>
    </professor>
</department>
```

Figure 6: **Redundancy in XML**

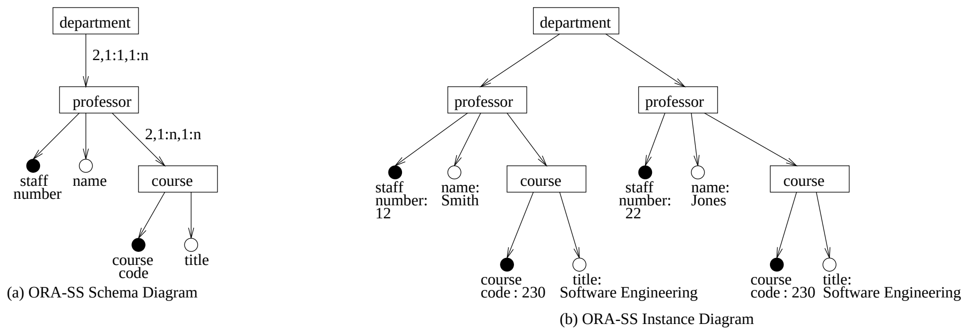

Figure 7: **Nested Object Classes**

Consider the corresponding ORA-SS schema diagram and instance diagram in Figure 7 for the XML document in Figure 6. There is a one-to-many binary relationship between *department* and *professor*, and a many-to-many binary relationship between *professor* and *course*. Note that the same course information is repeated for each professor that teaches the course. This redundancy can be avoided if *course* is referenced by *professor* rather than nested within *professor*, as shown in Figure 8. When the semistructured data is based on this logical model, the redundancy is eliminated (Figure 9).

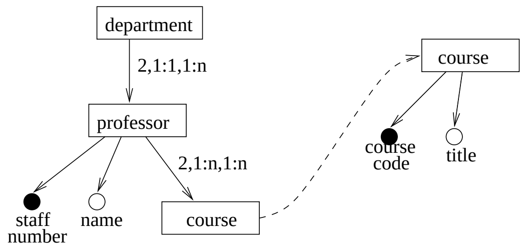

Figure 8: **Referenced Object Classes in an ORA-SS schema diagram**

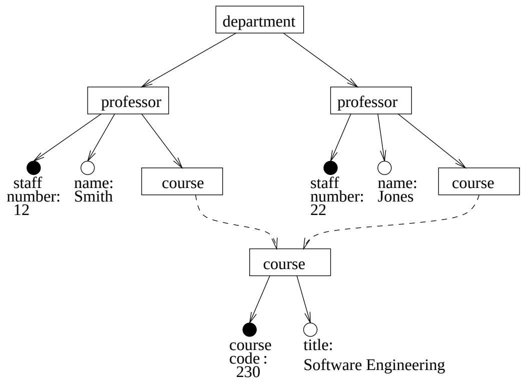

Figure 9: **Referenced Object Classes in an ORA-SS instance diagram**

In the example above the redundancy is very obvious, but there are more complex situations where the redundancy is harder to recognize.

3.2 Storage of Semistructured Data

In order to design an efficient and consistent organization of data in a data store, it is essential to have an algorithm that maps the logical data model to the data store. For example, the mapping algorithms from the ER data model to the relational data model. In this section, we outline an algorithm that maps ORA-SS schema diagrams to the object relational model. Such an algorithm demonstrates how semistructured data can efficiently and consistently be stored in a nested relational database management system like Oracle 8*i*.

Consider the schema diagram and associated object relational schema in Figure 10. The algorithm would include:

1. for each object class, create a (possibly nested) relation,

   (a) the key of the object class is the key of the relation,

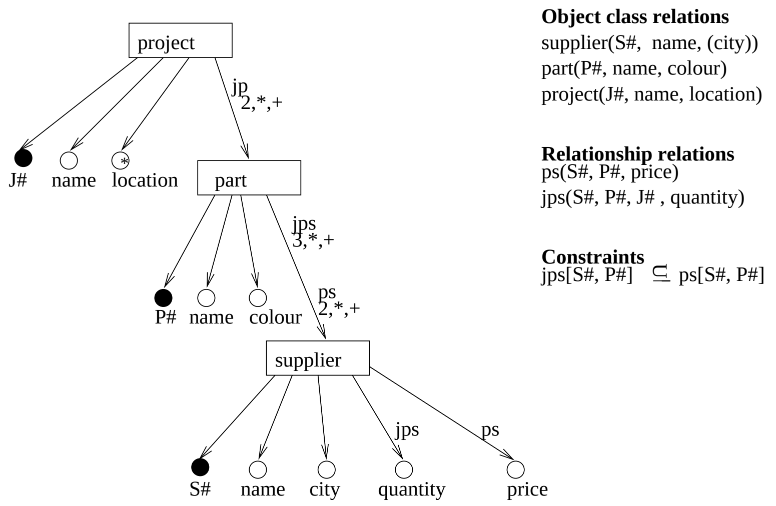

Figure 10: **Supplier, Part, Project Relationship Set Modeled using an ORA-SS schema diagram**

   (b) each single-valued attribute of the object class is a single-valued attribute of the relation,
   (c) each multivalued attribute is a set-valued attribute in the relation,
   (d) each reference is a foreign key in the relation.

2. for each relationship type, create a (possibly nested) relation,
   (a) each single-valued attribute of the relationship type is a single-valued attribute of the relation,
   (b) each multivalued attribute is a set-valued attribute in the relation.

Note that in contrast to existing models, ORA-SS enables the mapping algorithm to correctly associate the attribute price with part and supplier in the *ps* relation, and the attribute quantity with project, part and supplier in the *jps* relation.

3.3 Reachability and Defining Views

The labels on attributes and relationship types convey useful semantics. In particular, they show what is reachable from a referencing entity. An object class cannot reach attributes or relationship types that belong to relationships that they do not participate in. Consider Figure 11(a), where there is a reference from object class *member of sport club* to *student*. The object class *member* cannot reach attributes or relationship types that belong to relationships *cs* or *cst*. Instead, *member* can only reach attributes *student number*, *name*, *address*, and *years member* as shown in Figure 11(b). This information cannot be derived from DOM [9], or OEM [14] diagrams, where attributes of relationship types and the degree of relationship types are not expressed explicitly.

ORA-SS can also be used to distinguish between meaningful views and views that are not meaningful. Figure 12 shows four meaningful views of the schema in Figure 10. The attribute price is an attribute of the relationship type (*ps*) between supplier and part, and the attribute quantity is an attribute of the tertiary relationship type (*jps*) between part, supplier and project.

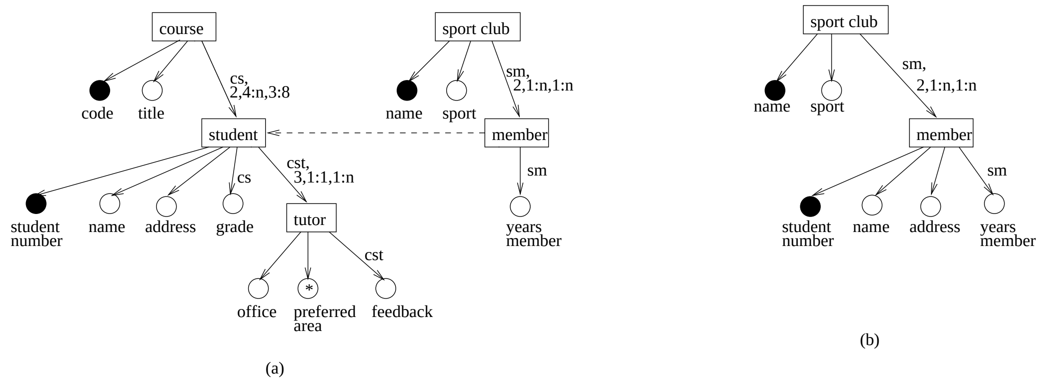

Figure 11: **Attributes and Relationship Types reachable from** ***sport club***

This is very obvious in the ORA-SS schema but is not obvious in the corresponding DOM or OEM diagrams, where the relationship types that attributes are related to are not shown explicitly.

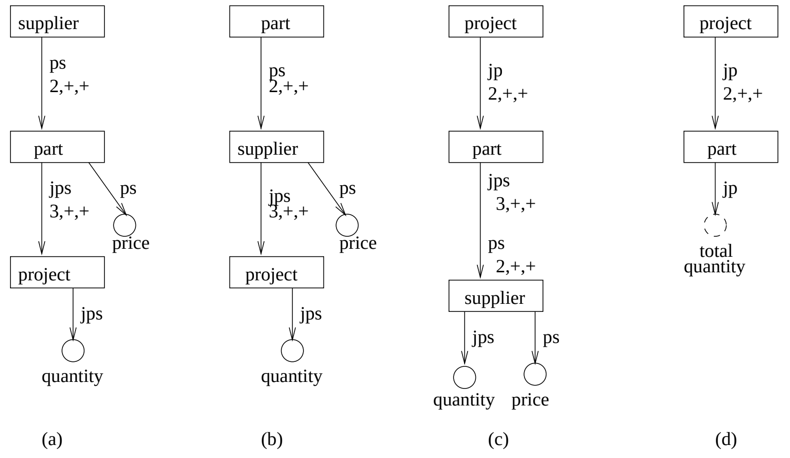

Figure 12: **Alternative Views of the Supplier, Part, Project Relationship**

This example can be generalized to the following heuristics:

1. The order of the object classes in a relationship type can be changed, and the attributes of the relationship types must be reorganized accordingly.
2. Object classes can be dropped from a relationship type. In this case, the attributes of the relationship types will be dropped or replaced by an aggregate of the attribute.

3.4 Evaluating XML Queries on an ORA-SS Databases

In Section 3.2, we describe how an ORA-SS schema diagram can be mapped to an object relational database. Suppose we had a mapping from an XML DTD or XML-Schema to an ORA-SS schema diagram, then it is necessary to define a way to query the data that is stored in the underlying database. We would expect the queries to be written in an XML query language, and that the results returned from the database would also be in XML. Evaluating XML queries against an ORA-SS database requires the following steps:

- Map XML schema to a ORA-SS schema diagram. Depending on how the XML schema is represented initially, this may involve human interaction to flesh out details that are not represented in the XML schema definition that need to be represented in the ORA-SS schema definition. For example, if the XML schema was represented as a DTD, then extra cardinality information would need to be added.
- Map the ORA-SS schema diagram to a storage schema. Such a mapping is described in Section 3.2.
- Map XML documents to ORA-SS database (in object-relational DBMS), using the above mappings.
- Map XML queries to queries on the ORA-SS database. There would be a need to handle recursion, negation, path expressions and wildcards.
- Construct a result from the database, that is in the form of an XML document.

The ability to represent the logical schema and the instances using ORA-SS diagrams helps enormously in defining the above procedure. The mapping defined in Section 3.2 results in an efficient storage of the data. The challenge would be to optimize the ORA-SS queries against the database.

3.5 Translating a Relational Schema to an ORA-SS Schema

XML is increasingly being adopted for information publishing on the World Wide Web. However, the underlying data is often stored in relational databases. Some mechanism is needed to convert the relational data into XML data. There has been research to efficiently translate relational schemas to XML [10, 18]. One of the major challenges is to find an effective way to generate an XML structure that is able to describe the semantics and structure in the underlying relational database.

Semantic enrichment of relational schemas model have been extensively studied in [3, 8, 11]. Functional dependencies and inclusion dependencies have traditionally been used to aid the translation of relational database into semantic data models such as the Entity-Relationship model and the object-oriented model. However, functional dependencies and inclusion dependencies are basically constraints to enforce the integrity of a database. [11] introduce the concept of semantic dependencies to represent the relationship between two sets of attributes at a semantic level.

We offer ORA-SS as a middleware to support the schema conversion from a semantically enriched relational schema to the semistructured XML. ORA-SS allows us to handle the translation of a set of related relations and distinguish attributes of relationship types from attributes of object classes, multivalued attributes, different types of relationships such as binary, n-ary, recursive and ISA. The structure of the XML data obtained is able to reflect the inherent semantics and implicit structure in the underlying relational database without unnecessary redundancy and proliferation of disconnected XML elements.

The main steps in the translation would consist of:

- Identifying object classes and their attributes from object relations.
- Identifying relationship types and their attributes from relationship relations.

- Identifying class hierarchy from the attributes of object relations.

4 Conclusion and Future Work

In the current data models for semistructured data it is not possible to model the kind of information that is traditionally needed when organizing data storage, designing repositories, or defining views. We have presented the ORA-SS data model in which this information can be represented. The data model consists of four diagrams: the schema diagram, the instance diagram, the functional dependency diagram and the inheritance diagram. The features of the schema diagram that make it the obvious choice for modeling semistructured data schemas include: the nested structure of the data is inherent in the schema diagram, a distinction is made between object classes, relationship types and attributes, and the relationship type attributes are distinguished from the object class attributes. The instance diagram shows the instances of object classes, relationship types and attributes. The functional dependency diagram shows the cardinality of object classes in relationships. The inheritance diagram shows commonality in attributes between object classes. Such information is lacking in existing semistructured data models.

In this paper, we have outlined how the ORA-SS data model can be used for recognizing redundancy in semistructured data, designing data stores for semistructured data and defining views on semistructured data. The cardinality of relationship types allows us to recognize possible redundant data. Permitting redundant data leads to update anomalies when the data is stored. We have given a mapping from ORA-SS to the object relational data model. Such a mapping is a central part of storing XML data in a database, and subsequent evaluation of XML queries against the data. We have also defined which attributes are reachable by an object class that references another object class and how semistructured data can be reorganized to give meaningful views of the data. ORA-SS schema diagrams can be used to determine which are meaningful and not meaningful views of the data.

The work presented in this paper can be used as the foundation for other work in the semistructured data field. We list some further work possibilities here:

- We have discussed how the ORA-SS data model can be used to recognize if there is redundancy in a semistructured data repository. Hence, normal forms for semistructured data could be defined based on the ORA-SS data model.

- We have shown how semistructured data and XML data can be mapped to the object relational data model. More work needs to done to design efficient storage organizations for other data models and defining access paths to the data.

- The ORA-SS data model can be used to identify which views are meaningful. This work can be extended to define how materialized views are updated, what updates to views are valid, and how updates to views are propagated to the underlying data.

- The ORA-SS data model provides a user-friendly way to visualize the instance and schema of a semistructured data store, and can be used in tools that require data visualization.

- Building tools to create web pages and to deal with semistructured data is a very active research area. The ORA-SS data model is the ideal notation on which to base such a tool, because the notation is simple and yet it describes the semantics that you need when defining the structure of the web page and how the data should be stored.

- The ORA-SS data model provides a simple and standard way for representing semantics which can be used for data integration. A large part of data integration involves finding equivalences or matches between two or more schema. The problem of finding equivalences between diagrams is very complex. It is easier to find equivalences automatically in a standard textual description. We have already started describing a syntax for the textual description.

- Using the data model to model real systems will show what extensions are necessary from a practical perspective.

The work described in this paper is very preliminary and from the list of further work, it is clear that the data model has a lot of potential. We are currently investigating some of the issues raised.

References

[1] Tim Bray, Jean Paoli, C.M Sperberg-McQueen, and Eve Maler (eds). Extensible markup language (XML) 1.0, second edition. http://www.w3.org/TR/2000/REC-xml-20001006, 6 Oct, 2000.

[2] Peter Buneman, Susan B. Davidson, Mary F. Fernandez, and Dan Suciu. Adding structure to unstructured data. In *Proceedings of the 6th International Database Theory Conference Database Theory (ICDT'97) Database Theory - ICDT '97*, volume 1186 of *Lecture Notes in Computer Science*, pages 336–350. Springer, 1997. There is also a VLDB journal 2000 version.

[3] M. Castellanous and F. Saltor. Semantic enrichment of database schema: An object-oriented approach. In *Proceedings of First Int. Workshop on Interoperability in Multidatabases*, 1991.

[4] Sophie Cluet, Claude Delobel, Jerome Simeon, and Katarzyna Smaga. Your mediators need data conversion! In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 177–188. ACM Press, 1998.

[5] Gillian Dobbie, Xiaoying Wu, Tok Wang Ling, and Mong Li Lee. ORA-SS: An object-relationship-attribute model for semi-structured data. Technical Report TR21/00, School of Computing, National University of Singapore, 2000.

[6] Mary Fernandez, Daniela Florescu, Jaewoo Kang, Alon Levy, and Dan Suciu. Catching the boat with Strudel: experiences with a web-site management system. In *SIGMOD*, pages 414–425, 1998.

[7] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *Proceedings of 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 436–445. Morgan Kaufmann, 1997.

[8] J-L Hainaut, M. Chandelon, and et al. Transformational techniques for database reverse engineering. In *Proceedings of 12th International Conference on ER Approach*, 1993.

[9] Arnaud Le Hors, Gavin Nicol, Lauren Wood, Mike Champion, and Steve Byrne (eds). Document Object Model (DOM) level 3 core specification. http://www.w3.org/TR/2001/WD-DOM-Level-3-Core-20010605, 5 June, 2001.

[10] Dongwon Lee, Murali Mani, Frank Chiu, and Wesley W. Chu. Nesting-based relational-to-xml schema translation. In *ACM SIGMOD Int'l Workshop on the Web and Databases (WebDB)*, May 2001.

[11] T.W. Ling and M.L. Lee. Relational to entity-relationship schema translation using semantic and inclusion dependencies. *Journal of Integrated Computer-Aided Engineering*, pages 125–145, 1995.

[12] Lotus development corporation. http://www.lotus.com/.

[13] B. Luddscher, R. Himmervder, G. Lausen, W. May, and C. Schlepphorst. Managing semistructured data with FLORID: A deductive object-oriented perspective. *Information Systems*, 23(8):589–613, 1998.

[14] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3):54–66, 1997.

[15] Tova Milo and Sagit Zohar. Using schema matching to simplify heterogeneous data translation. In *Proceedings of 24rd International Conference on Very Large Data Bases (VLDB'98)*, pages 122–133. Morgan Kaufmann, 1998.

[16] Jayavel Shanmugasundaram, Kristin Tufte, Chun Zhang, Gang He, David J. DeWitt, and Jeffrey F. Naughton. Relational databases for querying XML documents: Limitations and opportunities. In *Proceedings of 25th International Conference on Very Large Data Bases (VLDB'99)*, pages 79–90. Morgan Kaufmann, 1999.

[17] Tamino - An Internet Database System. http://www.tamino.com/.

[18] V. Turau. Making legacy data accessible for XML applications. http://www.informatik.fh-wiesbaden.de/ tarau/veroeff.html/, 1999.

[19] XML schema. http://www.w3.org/XML/Schema.