Chapter XVIII
Web Data Warehousing Convergence: From Schematic to Systematic

D. Xuan Le
La Trobe University, Australia

J. Wenny Rahayu
La Trobe University, Australia

David Taniar
Monash University, Australia

ABSTRACT
This paper proposes a data warehouse integration technique that combines data and documents from different underlying documents and database design approaches. The well-defined and structured data such as Relational, Object-oriented and Object Relational data, semi-structured data such as XML, and unstructured data such as HTML documents are integrated into a Web data warehouse system. The user specified requirement and data sources are combined to assist with the definitions of the hierarchical structures, which serve specific requirements and represent a certain type of data semantics using object-oriented features including inheritance, aggregation, association and collection. A conceptual integrated data warehouse model is then specified based on a combination of user requirements and data source structure, which creates the need for a logical integrated data warehouse model. A case study is then developed into a prototype in a Web-based environment that enables the evaluation. The evaluation of the proposed integration Web data warehouse methodology includes the verification of correctness of the integrated data, and the overall benefits of utilizing this proposed integration technique.
INTRODUCTION AND MOTIVATION

Currently, there are more and more techniques being provided to accommodate the high demand for exchanging and storing business information including Web and operational data. While the well-defined structured data are operated and stored in relational, object-oriented (Buzydlowski, 1998), object relational database environments, semi-structured data in XML or unstructured documents are stored in HTML. The problem of related information being separated and stored in multiple places happens quite often within an organization. Information from these applications is extracted and further developed into business analysis tools such as OLAP and data warehousing, which aim to support data analysis, business requirements, and management decisions.

Relevant business Web data have rapidly increased in significant amounts. Recently, XML has increased in popularity and has become a standard technique for storing and exchanging information over the Internet. The data integration (Breitbart, Olson, & Thompson, 1986) in the data warehousing has certainly received a lot of attention. There are three particular articles that are very close to the work in this article. Jensen, Moller and Pedersen (2001) allow an integration of XML and relational data. Even though the object-oriented concept is used in this model, the semantic contribution in this work lacks object-oriented features. Therefore, the semantics of data have been only partially supported. Other systems (Golfarelli, Rizzi, & Birdoljak, 1998, 2001; Huang & Su, 2001) focus on supporting Web data at the schematic level. While their initial focus is to incorporate XML data, Relational data have also been mentioned but not yet been incorporated. They mostly concentrate on the creation of a logical model.

Hence, it is clear that there is yet to be developed a standard integration technique that provides a means of handling multiple data sources being integrated into a data warehouse system (Bonifati, Cattaneo, Ceri, Fuggetta, & Paraboschi, 2001), and allowing a full capture of semantics of data in the data source models.

The purpose of this article can be summarized as follows:

- To ensure the integration technique allows a meaningful uniformed integrated object-oriented data warehouse structure.
- To ensure the integrated data and their semantics are explicitly and fully represented.

Figure 1. Integration Web data warehouse overview
Table 1. Categorization and summary of existing work

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Integration Methodology</th>
<th>Analysis and Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gupta and Mumick (1995)</td>
<td>Views</td>
<td>Map local source structures to global views to accomplish specific needs.</td>
</tr>
<tr>
<td>Calvanese et al. (1998)</td>
<td>Reasoning techniques</td>
<td>Rewrite queries procedurally to declare relationships between data sources structure and data warehouse structure.</td>
</tr>
<tr>
<td>Cabibbo et al. (1998); Gopalkrishman et al. (1998)</td>
<td>Relational star schema and goal-driven analysis</td>
<td>Specify user-requirements on a star schema. Apply goal-driven analysis for selecting information to the target schema.</td>
</tr>
<tr>
<td>Chen, Hong, and Lin (1999); Filho, Prado, and Toscani (2000); Mohamah, Rahayu, and Dillon (2001)</td>
<td>Object-oriented model</td>
<td>Lack semantic representations. Use only an aggregation modeling feature to represent the data.</td>
</tr>
<tr>
<td>Miller, Ho-navar, Wong, and Nilakanta (1998); Serrano, Calero, and Piattini (2005)</td>
<td>Mapping object views</td>
<td>Extensive views allow various levels mapping. Develop a prototype to materialized views.</td>
</tr>
<tr>
<td>Gopalkrishman et al. (1998).</td>
<td>Object-oriented model</td>
<td>Lacked semantic representations. Use only inheritance modeling features to represent the data.</td>
</tr>
<tr>
<td>Huynh, Mangisengi, and Tjoa (2000).</td>
<td>Object-oriented model and mapping object methodology</td>
<td>The reversible mapping from object to relational environment causes possible lost of data semantics.</td>
</tr>
<tr>
<td>Jensen et al., (2001).</td>
<td>UML model</td>
<td>Lack of data representation showing only aggregation relationship. Address both XML and relational data.</td>
</tr>
<tr>
<td>Byung, Han, and Song (2005); Nummenmaa, Niemi, Niinimäki, and Thanisch (2002)</td>
<td>Relational star schema</td>
<td>Enable query to distribute XML data in OLAP database Address only XML data.</td>
</tr>
<tr>
<td>Nassis, Rahayu, Rajugan, and Dillon (2004)</td>
<td>UML Model</td>
<td>Specify user-requirement and XML structures on the object-oriented model.</td>
</tr>
</tbody>
</table>

- To ensure a proposed integrated data warehouse system with consistency and high quality.
- To ensure the correctness of integrated data and benefits such as usefulness of the proposed integrated data warehouse system.
Figure 1 shows an overview of the proposed works in this article. The integration technique starts with a conceptual integrated data warehouse model (Ezeife & Ohanekwu, 2005) where the user requirement and underlying data source structures are used to assist with the design. The integrated Web data warehouse conceptual model deals with class formalization and hierarchical structures. The specified conceptual integrated Web data warehouse model has created a need for an integrated Web data warehouse logical model where underlying source structures are then absorbed and specified onto the existing conceptual Web integrated Web data warehouse model. The proposed Web integrated data warehouse models are then translated into a suitable implementation format, which enables a prototype to be developed.

In order to confirm the efficiency of the proposed integration technique, a verification of integrated data is for the purpose of confirming the correctness and quality in the integrated data. This is done so that for each query requirement, a query is issued to access the integrated data warehouse system, and a set of queries access independent systems. The result that is obtained by the query that accessed the integrated data warehouse system is equivalent with the accumulative result that is obtained by queries that access one or more data source systems. The verification of the result would confirm the correctness and consistent quality of the integrated data alone, and the integration technique in general.

A SURVEY OF EXISTING DATA WAREHOUSE INTEGRATION APPROACHES

The existing approaches are classified into three categories. Table 1 briefly summarizes the existing approaches by category.

Category 1 includes the existing integration technique that can integrate only relational data into a data warehouse system. A data integration problem solved by proposing two approaches, namely, declarative and procedural can be found in the works of Calvanese, Giacomo, Lenzerini, and Rosati, (1998) and Lenzerini (2002) where as Cabibbo and Torlone (1998) and Gopalkrishman, Li, and Karlapalem (1998) propose different techniques to integrate data that are based on the requirements gathered from the user specification and also from studying the conceptual design of the operational source data. In order to create the model, a matching of requirements to sources is needed before creating fact and dimensions.

Category 2 shows techniques for handling complex information, which are different from the techniques that handle simple data types, which are available in the relational database. An object data warehouse approach allows an integration of both simple and complex data types. Its main function is to accomplish all important object-oriented concepts and additional features such as object ID and persistent object handling.

An object-oriented model extends the technique to handle the transition from relational data to object data (Filho et al., 2000; Gopalkrishman et al., 1998; Hammer, Garcia-Molina, Widom, Labio, & Zhuge, 1995; Huynh et al., 2000). However, the proposed model lacks a utilization of object-oriented features that result in insufficient representation of the semantics. Miller et al. (1998) introduce an object view in the mapping technique. They adopted the extensive view system to create views. However, views creation depends on the number of base classes.

Category 3 has allowed the data integration to move on to an advanced level where XML data is the main motivation. Web data nowadays can easily be found in XML structure, which has many possibilities for data modeling. This is because XML is well designed to support object-oriented modeling concept; the data semantics are very rich. Therefore, techniques for integrating XML data into a data warehouse system (Nassis et al., 2005; Rusu, Rahayu, & Taniar, 2004, 2005)
needs to take more cautious because unlike relational and object data, XML data are classified as semi-structure.

While Golfarelli et al. (2001) try to deal with DTD and XML schema, Jensen et al. (2001) propose query to distribute XML data to an OLAP database according to the data representation. Part of our work is very much similar to the work of Jensen et al. (2001), we consider both XML and relational data for integration, and we also combine user requirements and underlying data structures to assist with the design. The difference between our work and the rest is that now we are handling three categories simultaneously. Not only are relational and XML data being considered, we also consider object data and other Web data structure such as HTML.

**PROBLEM DEFINITION AND BACKGROUND**

**Identified Problems**

**Schemas**

The most popular existing model in data warehousing is the star schema. The star schema allows business requirements to be organized and represented in a fact and dimensions surrounding fact. Dimensions are modeled on a flat level; therefore, it limits the data representations for both relationships and business requirements.

Unlike the star schema, the snowflake or star flake schema provides modeling of hierarchical relationships within the dimensions. The existence of hierarchies in the dimensions stores the whole attribute hierarchically and shows only one type of relationship, which is association. While it improves on the modeling representation, it creates more data-model complexity and therefore introduces implemental complexities.

The integration of the real world problems can be represented in a multidimensional model that consists of dimensions and fact using the hierarchical concept. Allowing for hierarchies in the dimensions would reduce the complexity of snowflake and star flake to a more efficient and clean integrated model while still being able to achieve a full data semantic capture.

**Data Retrieval**

The translation of the integrated data warehouse model into an implementation-ready format aims to address the adaptation of the object-oriented modeling concept into an implementation database environment where both object data and relational structures are maintained. Retrieved information must be correct and consistent in this proposed implementation when complex queries are specified in OLAP components. Performance of complex queries must be achievable in an efficient data accessing manner against the existing complex queries of the existing systems.

**Background**

We adopt object-oriented features, a semantic network diagram, and the TKPROF utility to assist with our strategy for solving the problem. They are briefly described as follows:

- **Object-oriented design concept:** The powerful features have allowed a problem to be modeled in much better semantics representations. Collection type allows the multi-values attribute to handle the storing of data in a more efficient manner using ROW, SET, and ARRAY. Features like aggregation allow a whole problem to be modeled as “part-of” where a lower hierarchy is part of the upper one, or part can be an existence-dependent or existence-independent.

  When the part is considered as existence dependent, it means that the part cannot be shared with other classes or removed from the whole.
Whereas, Existence independent is where the part can be shared with other classes and can be removed independently of the whole.

An inheritance (Rahayu, 1999; Rahayu, Chang, Dillon, & Taniar, 2000) type is where the problem is modeled as a super class with sub-classes. The sub-class utilizes the information in the super-class and its own information to specialize itself.

An association relationship represents a connection between two objects. There are three types of association relationships such as one to one, one to many, and many to many. The type being used depends on the criteria of the problem.

- **Semantic Network Diagram:** If given an XML document as one of the data sources, we employ the semantic network diagram (Feng, Chang, & Dillon, 2002) to translate XML data into the proposed integrated model. The semantic network diagram is divided into the semantic level and schema level. The former developed a specific diagram from the XML document structure and the latter maps from this specific diagram into the target model, an integrated data model. The semantic network diagram is divided into four major components: nodes, directed edges, labels, and constraints. Suppose a semantic network diagram in Figure 1 is studied.

Based on the construction rules to formalize a semantic network diagram (Feng et al., 2002; Pardede, Rahayu, & Taniar, 2004), there are five nodes: A, B, X, Y, Z in the diagram. The first two nodes are the complex nodes while the rest are the basic nodes. There are four directed edges representing the semantic relationships between the objects. In our work, we use different labels to indicate the relationship corresponding to each edge. Different labels are interpreted as follows:

- \( p \) indicates “in-property”; \( g \) indicates generalization; \( a \) indicates aggregation; \( c \) indicates composition.

Various types of constraints such as uniqueness, cardinality, ordering, etc., can also be added to the nodes or edges. The modeling representation in Figure 2 presents a well-defined conceptual design from XML data. The attributes or elements declarations and simple or complex type (Pardede, Rahayu, & Taniar, 2005) definitions in XML schema are mapped into the four components or directed edges.

**INTEGRATION PROPOSED TECHNIQUE**

The structures of underlying data sources can be the combination of relational structures and structures that are available in XML documents and object databases.

- **Translation Technique of HTML Data into XML Structure:** Before conducting the integration of a Web data warehouse model, we adopt the mapping tool and technique that is proposed in the works of Bishay, Taniar, Jiang, and Rahayu (2000), and Li, Liu, Wang, and Peng (2004) to map from HTML data to XML data so that attributes can be identified. Figure 4 shows HTML data that are translated to XML schema using very basic and straightforward mapping steps. More information on the mapping and transforming techniques can be found in these two references.

1. **Mapping Rule:** Referring to Figure 3, let the content of table XYZ is a set of rows \(<TR>\) and each row contains a set of column \(<TD>\); XYZ is mapped to an XML schema structure; \(<TR>\) is mapped to the \(<xsd:
Sequence>; <TD> is mapped to the <xsd:element> within the sequence.

2. **Motivation by a Case Study:** To provide a feasible example for this article, we illustrate the proposed approaches based on the need to build a data warehouse system for university enrolments. Information about the enrolments is stored in relational and Web forms. This is due to the fact that each individual faculty uses its own system and none is currently linked.

One faculty might have its own Web-based system while the others, for various reasons, might have just a normal database system to handle the enrolment of students. It is the goal of the university to construct a data warehouse system in order to analyze student enrolments in areas/subjects/degrees, and also the trend of enrolments in different years including semesters. The university is also interested in the analysis of degree enrolments for a particular area; for example, for the Masters degree, there might be more students enrolled in coursework than in research. In some rare cases, a university may be limited in its ability to provide both research and coursework. Thus, it is interesting to see the relationship between these parties. A faculty may be formed by one or more schools, and a certain number of degrees belong to a particular school. A study of an advanced subject is required for some prerequisites. The university would like information about the prerequisites to be kept in the warehouse system for future analysis. Points to consider are that a specific degree belongs to only one faculty. A subject can be attended by students across the degrees.

The methodology for specifying the conceptual integrated data warehouse model in two phases is as follows: phase (a) consists of the steps, which are temporarily referred to as *conceptual defined sequence*, to assist with the process of creating the conceptual integrated dimensions and fact; phase (b) is an extension of phase (a) to allow data structures of relational and HTML/XML data sources to be fully unified and incorporated in the integrated data warehouse model.

**Conceptual Web Integrated Dimensions and Fact**

Conceptually, starting with the assumptions of the user specified requirements and information related to underlying sources in relational and XML, we form a set of steps for defining our integrated Web data warehouse model. Please note that by this time, HTML data have been translated to XML structure. The methodology consists of the following steps, which we temporarily refer to as a *conceptual defined sequence*, to assist with the process of creating the model:

1. **Simplifying the requirements:** Structures of underlying data sources can also be simplified if possible.
2. **Defining integrated dimensions involves two sub-steps:** (a) Specifying n classes where \( n \geq 1 \); (b) classifying hierarchy: additional specified information by any other means is a great advantage. Suppose two classes A and B in a dimension, the relationship between A and B can either be a, b, or c.

   a. **Aggregation:** Deals with the dependence between the classes. Considering the cardinality where needed, -to-one or to-many, between the base classes and sub-classes.

   b. **Inheritance:** Categories subtypes and super-types.

   c. **Collection:** Handles multi values in an attribute. This relationship in our approach is not for hierarchy building, but rather for storing data in a more efficient manner.

   d. **Association:** Is when two classes have an association relationship, using a -to-one; -to-many to describe the association between classes.

3. Defining Fact: A simple, single fact, which is surrounded by integrated dimensions. Hierarchy and cardinality should be identified.

The **conceptual defined sequence** is now used to specify the conceptual integrated Web dimensions and fact as follows:

- **Inheritance Type Dimension:** Dimensional analysis is such “…The university is also interested in the analysis of degree enrolments for particular type, for example, for a Masters degree, there may be more students enrolled in course work than in research but it may be that a university has a strong constraint in providing both research and coursework…” applying the **conceptual defined sequence**, a conceptual degree is specified as follows:

1. Simplifying requirements. A Degree can be further classified as a Research Degree or a Coursework Degree.

2. Identified Dimension {Degree}

**Classes {Degree, Research, Coursework}**

Hierarchy {Generalization} **additional formation:** the same number of years applies to all Masters degrees. Extra information is needed to support...
the specialization of a degree type. An inheritance type is an ideal modeling feature because a degree is a generalization and research or coursework is specialization. No cardinality.

A conceptual degree dimension is derived based on steps 1 and 2 shown in Figure 4.

- **Collection Type Dimension:** Dimensional analysis may be: “...A study of an advanced subject is required for some prerequisites. The university would like information about the prerequisites to be kept in the warehouse system for future analysis...,” applying the conceptual defined sequence; a conceptual degree is specified as follows.

  1. Simplifying requirements. A subject needs to store its prerequisites. Each subject has two prerequisites at most.
  2. Identified Dimension {Subject}
     Classes {Subject}
     Hierarchy {NIL} A collection type is an ideal modeling feature because it allows a prerequisite to be modeled as an attribute that stores multi-values using array, row, set. No cardinality.

A conceptual subject dimension is derived based on step 1 & 2 shown in Figure 5.

- **Aggregation Type Dimension:** As recalled earlier, we claim that aggregation is further grouped into two groups: Non-shareable-existence dependent and shareable-existence dependent.

- **Non-shareable Existence Independent Type Dimension:** Dimensional analysis is such “...A faculty may be formed by one or more schools and a certain number of degrees belongs to a particular school...,” applying the conceptual defined sequence, a conceptual faculty is specified as follows:

  1. Simplifying requirements. A faculty can own none or more than one school.
  2. Identified Dimension {Faculty}
     Classes {Faculty, School}
     Hierarchy {Aggregation} additional formation: a Faculty can be existed without a School. One-to-many.

A conceptual faculty dimension is derived based on information above, shown in Figure 6.

- **Shareable Existence Independent Type Dimension:** Dimensional analysis is such “...also the trend of enrolments in different years including semesters ...,” applying the conceptual defined sequence, a conceptual time is specified in Figure 7.

  1. Simplifying requirements. A time can also include semester. Semester is needed for enrollment.
  2. Identified Dimension {Time}
     Classes {Time, Semester}
     Hierarchy {Aggregation} additional information: A semester can be shared with other classes. Time has many months or years. And a year has more one or more semesters.

Thus, it is a many-to-many as shown in Figure 7.
• **Fact Class**: Fact analysis is such "...compute student enrolment to timely analyze the trends and performance of subjects and degrees in faculties..." From item 3 in section A, we have Class{Uni_Fact}; Hierarchy {Association}; one-to-many.

A conceptual fact class is derived in Figure 8 surrounding the support of the conceptual integrated dimensions:

**Logical Web Integrated Dimensions and Fact**

In this section, the rest of the integrated dimensions and facts are specified in greater detail to directly utilize the structures of underlying sources. It assumes that both relational data sources and HTML/XML documents are retrieved based on the user requirements and available structures in the sources.

• **Adding Attributes to Collection type Dimension**: A Semantic network diagram has not yet formalized a representation for a collection type. Thus, we propose a "C" label indicating a collection type that represents a semantic in the data complex type.

With reference to Figure 9, shows *relational data & semantic network diagram* Attrs{A, B, M1, M2..Mn} are simple data types; Attrs{M1, M2} are multi-valued attributes in relational table; And Attrs{M1, M2} sub-elements in Semantic Network Diagram; ComplexType (Type 1, Type 2). Adding attributes to a collection type dimension consists of two steps:

• **Step 1**: For a relational data source table that has attributes {A, B, M1, M2}, which are required for analytical information. Attribute {A, B} are added to Dimension 1. Attributes { M1, M2} are stored as a {C} attribute that has a VARRAY type. Attribute {C} is an array type that take two elements, which is also added to Dimension 1.

• **Step 2**: For two complex types namely Type 1 and Type 2 with elements {A, B} and {M1, M2} respectively, Type 2 is an inner complexType element in Type 1. Type 2 element contains sub-elements {M1, M2}. Thus, element {A, B} in Type 1 are mapped to attributes {A, B} in Dimension 1; sub-element{M1, M2} are mapped to an element
Example: Conceptual subject dimension in Figure 5 is now presented here to add appropriate attributes and data structures in order to complete the integration of a logical integrated subject dimension shown (Figure 10).

Step 1: For subject relational data table provided by health and science faculty with a set of attributes {Subjectid, Subjectname, Req1, Req2}, which are required for analytical information. Attributes {SubjectID, Subjectname} are added to the conceptual subject dimension. Attributes {Req1, Req2} are stored in a VARRAY element {Prerequisites}, which can take two elements in a single record. Attribute {Prerequisite} is then also added to subject dimension. Refer to SubjectDimension in Figure 10.

Step 2: For an outer complex type, SubjectType and elements {Subjectid, Subjectname, Refsubject}, {Refsubjectprereq} is an inner complexType element of SubjectType. Refsubject complexType contains sub-elements {Req1,Req2}. Thus, elements {Subjectid, Subjectname} in SubjectType are mapped to attributes {Subjectid, Subjectname} in SubjectDimension, which are added in step 1. Elements {Req1, Req2} are mapped to element{Reprequisite}. And element {Reprequisite} can contain up to two sub-elements as formed in step 1.

A complete subject integration forms classes and attributes as follows:

Figure 8. Conceptual fact surrounded by integrated dimensions
SubjectDimension (SubjectID, Subjectname, prerequisite<Varray>)

where SubjectID is primary key (OID)

- Adding Attributes to Inheritance Dimension: Figure 11 shows that relational data and semantic network diagram Attrs{A, B, D, E, F} are simple data types; Attrs{D} is a type attribute; generalized attributes{A,B} specialized Attrs{E, F}; ComplexType(Type 1, Type 2, Type 3...Type n}. Adding attributes to an inheritance dimension consists of two steps:

**Step 1:** For a relational data source table that has attributes {A, B, D, E, F}, which are required for analytical information. Dimension 2 is a super-type, which has one or more sub-dimensions. Each sub-dimension has one or more specialized attributes. To complete an integration of inheritance dimension: add specialized attributes{E,F} to each sub-dimension.

**Step 2:** For three complex types, namely Type 1, Type 2 and Type n with elements {A, B, D, E, F} are required analytical information. Type 1 is the base type where Type 2 and Type n are of the extension based Type 1. Element {A, B} in Type 1 are mapped to attributes {A,B} in Dimension 2. Extension base types Type 1 is mapped to sub-type of Value31; whereas Type n is mapped to Value32 respectively. An element such as {E} or {F} is mapped to its own class where appropriate.

**Example:** Conceptual degree dimension, in phase (i) Figure 4 earlier, is now presented in Figure 12 to add appropriate attributes and data structures in order to complete the integration of degree dimension.

**Step 1:** For a relational degree source table that has attributes {DegreeID, Degreename, Degreetype, Area, Major}, which are required for analytical information. DegreeDimension is a super-type which can have two sub-dimensions, research, and coursework. Each sub-dimension has one or more specialized attributes such as {Area} or {Major}. To complete an integration of the inheritance DegreeDimension: add generalized attributes{Degreeid,Degreename} to DegreeDimension; mapping Research value of DegreeType to Research sub-type and Coursework value of DegreeType to Coursework sub-type; Area is an attribute to specialize the research degree and major is the attribute to specialize coursework degree. Thus, attribute {Area} is added to Research sub-type and {major} is added to Coursework sub-type.
**Step 2:** For three complex types, DegreeType, ResearchType and CourseworkType with elements {DegreeID, DegreeName, Area, Major}. DegreeType is the base type where ResearchType and CourseType are of the extension base DegreeType. Element {DegreeID, DegreeName} in DegreeType are mapped to attributes {DegreeID, DegreeName} in DegreeDimension. ComplexType of Research of extension base DegreeType is mapped to sub-type of Research; whereas ComplexType Coursework is mapped to sub-type Coursework. Element such as {Area} and {Major} is mapped to its own Research and Coursework respectively.

A complete degree integration forms classes and attributes as follows:

- **DegreeDimension** `{DegreeID, DegreeName, prerequisite<Varray>}`
- **Research** `{Area}`
- **Coursework** `{Major}`

where DegreeID is primary key (OID)

- **Adding Attributes to Aggregation Dimension.**

**Non-shareable Existence Dependent type** is applied to a problem where “parts” are dependent on the “whole.” When the whole is removed, its
parts are also removed. With reference to Figure 13, Attrs{A, B, C, D, E, F} are simple data types; ComplexType (Type 1, Type 2). Adding attributes to aggregation dimension consists of two steps:

**Step 1:** For a relational data table 1 and relational data table 2 that have attributes {A, B, D, E}, which are required for analytical information. Relational data table 1 has a one-to-many relationship with relational data table 2. And relational data table 2 is composed of relational data table 1. Thus, relational data table 1 is a parent of relational data table 2.

**Step 2:** For two complex types namely Type 1 and Type 2 with elements {A, B} and {E, F}. If Type 2 is composed by Type 1 then Type 1 is mapped to Dimension 3 and element {A, B} in Type 1 are added to attributes {A, B} in Dimension 3. Type 2 is also mapped to component of Dimension 3 and elements {E, D} are added to Component of Dimension 3. Note that element names in Type 2 are not matched with element names in Component. For the time being, presumably element {E} is matched with element {E} and element {D} is matched with element {F}.

**Example:** Conceptual faculty dimension, section (A) Figure 6 earlier, is now presented in Figure 14 to add appropriate attributes and data structures in order to complete the integration of the degree dimension.

**Step 1:** For the relational faculty data source table and relational school data tables that have attributes {FacultyID, Facultyname} and {SchoolID, Schoolname}, which are required for analytical information. Relational Faculty has a one-to-many relationship to the Relational School Table. And the Relational School Table comprises the Faculty Table. Thus, the Faculty Table is a parent of the School Table. On the other hand, FacultyDimension and SchoolComponent have a Part-Of relationship. On the other hand, the FacultyDimension is a parent of SchoolComponents. The SchoolComponent is a non-shareable part which means that when FacultyDimension is removed, SchoolComponents is also removed. To complete an integration of FacultyDimension: add attributes {FacultyID, Facultyname} in Faculty Relational table to FacultyDimension; add attributes {SchoolID, Schoolname} in Relational School table to the corresponding SchoolComponent.
Step 2: For two complex types, namely Faculty type and School type with elements {FacultyID, Facultyname} and {SchoolID, Schoolname}. If the School type comprises the Faculty type, then Faculty type is mapped to FacultyDimension. The elements {FacultyID, Facultyname} in Faculty type are added to attributes {FacultyID, Facultyname} in FacultyDimension. School type is also mapped to SchoolComponent and elements.
{SchoolID, Schooname} in School type are added to SchoolComponent.

A complete faculty integration forms classes and attributes as follows:

FacultyDimension \{FacultyID, Facultyname\}
SchoolComponent \{SchoolID, Schoolname\}

where Faculty, SchoolID are primary keys (OIDs)

Shareable Existence Independent Type is applied where parts are independent of the whole. When the “whole” is removed, parts still remain.

The time conceptual dimension in Figure 7 now shows the process of specifying/mapping attributes in Figure 15. A complete time integration forms classes and attributes are formed.

PreferredSize \{TimeID, Description\}
Semester \{SemesterID, Description\}

where TimeID and SemesterID are primary keys

• **Adding Attributes Fact Class**

Figure 16 shows fact that is referenced to SubjectDimension, DegreeDimension, FacultyDimension and TimeDimension. Thus, keys of SubjectID, DegreeID, FacultyID and TimeID are
Figure 14. Adding/mapping attribute data conceptual faculty dimension

```
<xs:complexType name="FacultyType">
  <xs:sequence>
    <xs:element name="FacultyID" type="xs:ID"/>
    <xs:element name="FacultyName" type="xs:string"/>
    <xs:element ref="School" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="SchoolType">
  <xs:sequence>
    <xs:element name="SchoolID" type="xs:ID"/>
    <xs:element name="SchoolName" type="xs:string"/>
    <xs:element ref="Degree" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="School" type = SchoolType />
```

added to fact. Enrolment attribute is an aggregated value that is tracked by subject, degree, faculty, and times basis.

Figure 17 is a complete integrated university logical data warehouse model that is fully integrated with powerful object-oriented features where storing the hierarchy in a dimension table allows for the easiest browsing of subject, degree and faculty dimension data on yearly basis. In Figure 17, we could easily choose a class and then list all of that category’s sub-classes. We would drill-down into the data by choosing a specific type of degree and a general degree.

**DISCUSSION AND ANALYSIS**

We have successfully proposed an integration technique that allows data from different database design approaches to be integrated into a data warehouse model. The proposed data warehouse integration approach is clearly and simply pre-
Figure 15. Adding/mapping attribute data conceptual time dimension


<table>
<thead>
<tr>
<th>SubjectID</th>
<th>DegreeID</th>
<th>StudentID</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSE21AI</td>
<td>MCSM</td>
<td>14312345</td>
<td>12/02/1999</td>
</tr>
</tbody>
</table>


```xml
<xs:complexType name="EnrolType">
  <xs:sequence>
    <xs:element name="RefDegree"
      type="xs:IDREF" minOccurs="0"/>
    <xs:element name="RefSubject"
      type="xs:IDREF" minOccurs="0"/>
    <xs:element name="Dateenroll"
      type="xs:date" minOccurs="0"/>
    <xs:element name="SemesterID" type="xs:positiveInteger"
      [omitted elements for StudentNo.]
  </xs:sequence>
</xs:complexType>
```

Figure 16. Specifying attributes to university fact class

<table>
<thead>
<tr>
<th>Uni_Fact</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubjectID</td>
</tr>
<tr>
<td>DegreeID</td>
</tr>
<tr>
<td>TimeID</td>
</tr>
<tr>
<td>FacultyID</td>
</tr>
<tr>
<td>Enrolments</td>
</tr>
</tbody>
</table>

hierarchical dimensions aim to allow the easiest information browsing of the data detail at the lowest level possible. The complete range of data source models from different database design approaches, including well-structured data and semi-structured data, have been successfully integrated in the proposed model. Most importantly, large amounts of Web data such as HTML/XML are now formally handled in the proposed integrated data warehouse. The correctness of the proposed integrated data warehouse model is going to be evaluated in the next section.

**EXPERIMENTATIONS AND ANALYSIS RESULTS**

This section describes (i) the implementation of the proposed integrated data warehouse solution, using the case study, into a Web-based integrated data warehouse prototype. The prototype is then allowed us to carry out the analyses, which perform
Figure 17. Object-oriented integrated data warehouse logical model

Figure 18. System architecture overview

a validation of the correctness and consistency in the integrated data.

**System Implementation**

**System Architecture**

Figure 18 is an overview of the implementation that aims to create an integrated data warehouse system in which the front-end is the Web-based language to assist with the dimensional presentation and user interaction; the back-end is the Oracle 10g, which is used to store the physical database. While the front-end Web-based component allows the requests from the users and displaying of the results in dimensional forms, the back-end processes the user’s requests in SQL statements and returns the retrieved results to the front-end platform for manipulation.
Data Source Creation includes the relational data source and XML data source, while a relational data source system for a faculty is developed including SQL for tables creation and sources generation. Source generation is simplified with the assistance of store procedures and functions. The XML data source system is developed including SQL/XML (Loney & Koch, 2000; Melton, 2003) with table creation and source generation. The source generation includes XML Spy for creating and validating the XML schemas. XML documents are then generated and validated using the created XML schema, both being done using XML Spy tool. The XML schema for validation is stored in Oracle and XML documents that contain XML data sources, and are then loaded into an XML database.

Integrated Data Warehouse Schema Creation and Data Source Loading consist of SQL queries to assist with tables and object types creation for hierarchical Dimensions and Fact classes that build an object relational data warehouse schema in SQL *PLUS.

Following each dimension and fact creation, a stored procedure is written for transforming data sources from a relational system and XML system into the integrated dimensions and fact table and types. The stored procedures use object relational (Taniar, Rahayu, & Srivastava, 2003) syntax for inserting data as object types into relational tables.

The transformation of data into the target integrated data warehouse schema is a critical task because of the data's multiple sources, for example, a subject may be taken by the degree students in different faculties. Therefore, the merging of the data may result in conflicts of subject information. An attempt to overcome the conflict problems that may cause abnormal system behavior and inconsistency in integrated data quality, our in-

Figure 19. Result based on integrated data
Figure 20. Result based on data in multiple source systems

```sql
SELECT d.degreeid, TO_CHAR(e.e.date, 'yyy'), COUNT(*)
FROM Faculty e, enrollments e, school s, degree d
WHERE e.facultyid = s.facultyid
AND s.schoolid = d.schoolid
AND d.degreeid = e.degreeid
AND e.e.date = 'yyy';
```

Figure 21. Result based on integrated data

```sql
SELECT distinct ExtractValue(value(1), 'Degree/degreeID') as DegreeID,
10 TO_CHAR(ExtractValue(value(2), 'Enrollment/Enrollment'), 'yyy')
AS Enrollment,
11 ExtractValue(value(3), 'Enrollment/Enrollment') as Enrollment
FROM Expr
WHERE ExtractValue(value(1), 'Degree/degreeID') = 'ARTA'
GROUP BY ExtractValue(value(1), 'Degree/degreeID'),
ExtractValue(value(2), 'Enrollment/Enrollment'),
ExtractValue(value(3), 'Enrollment/Enrollment');
```
tegration technique also includes the checking of existing duplicate information and incorporation of techniques to handle such problems.

**Analysis Results**

Our query experimentation structure includes the validation of integrated data for ensuring the quality and consistency in the integrated data.

**Category 1: “The University is interested in the number of student enrolments in coursework degree on a yearly basis.”**

As processed by the query for criteria in Category 1, the result shown in Figure 19, we recall that a degree has been modeled using an inheritance object-oriented feature. In this object relational integrated data warehouse query that...
associates with the information of a specific degree type, it demonstrates the capture of information at the lower levels in the WHERE clause, in which the IS OF type is introduced to access the lower level’s information without needing to use a join operation between the degree and its subtype, coursework, based selection in the object relational syntax.

Once the selection of values is made, the query processing next applies the aggregate function SUM on the Enrolment values and GROUP BY the coursework degree further by year.

Figure 20 shows the first SQL statement that accesses the relational system, which produces 34 enrolments for degree ARTB in year 2001. The second SQL/XML statement accesses the XML system, which produces one enrolment for degree ARTB in year 2001. The sum of results in first two statements for degree ARTB in 2001 is the 35. ARTB degree in year 2001 is 35 enrolments in Figure 19 and Figure 20 (sum of results).

We may conclude that data have been consistently integrated in our integrated data warehouse system.

**Category 2:** An analytical requirement, which it demonstrates the ordering of the data result using rank function. “Retrieve the enrolments in the degrees group by each year and display only the top four rows from the ranked results which have the most enrolments in the degrees and year.”

As processed by the query for criteria in Category 2, the result shown in Figure 21, the query for this criteria falls into an advanced OLAP query category. Apart from foundation syntax such as SUM() and GROUP BY which have been used, the RANK() function has also been used to handle the ranking concept based on the total measurement in the SQL statement. The Ranking () function assigns ranks from 1 to N and skips a rank in case of ties. Therefore, if two degrees have the same enrolments, a tie ranking is applied to these two degrees. This is illustrated by the example below:

<table>
<thead>
<tr>
<th>Row</th>
<th>Rank</th>
<th>Enrolments</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>2678</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2678</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2778</td>
</tr>
</tbody>
</table>

As it can be seen, the result in Figure 21 that within a degree there are no enrolments in particular years, for instance the data of enrolments for the ARTA degree in 1994 is 176 enrolments and no enrolments in years 1995 and 1996. Now the question is: how many enrolments are there for ARTA degree in the year 1994 available in the source systems? Let’s us prove the number of enrolments in degree ARTA in 1994 with a further experimentation. The verification involves an SQL statement to retrieve all the data in the relational data source system and an SQL/XML statement retrieves all data in the XML document as demonstrated in Figure 22.

Figure 22 shows the first SQL statement that accesses the relational system, which produces the coursework degrees grouped by year with their associated number of enrolments. The second SQL/XML statement accesses the XML system, which produces the 176 enrolments for degree ARTA in year 1994. ARTA degree in year 1994 is 176 enrolments in both Figure 21 and Figure 22. We may conclude that data have been consistently integrated in our integrated data warehouse system.

**Discussion and Analysis**

The proposed integrated data warehouse technique explicitly provides a fully integrated solution that includes a conceptual design extended to the logical design and an implementation for verification. Our integration approach is divided into phases, which are engaged with a very simple set of instructions to assist with the establishment of integrated fact and dimensions. The dimensions may have hierarchies that have been handled with a range of object-oriented features.
From the implementation experimentation, the system has been successfully developed with full relational data and XML data loaded into independent source systems. These source data are then integrated using stored procedures for extracting, transforming, and loading into the target system.

Query processing and data retrieval have been considered more efficient because of the leveling data representation. In the completion of query processing, the complex queries have utilized analytic functions, as demonstrated in the analysis results section, to successfully retrieved integrated data.

For a validation of the correctness and consistency of our integrated data, we perform two sets of queries processing—(i) a complex query is issued to access the integrated data system to retrieve the first result; (ii) and a set of queries to access the independent source systems to retrieve the one or more results and the sum of these results should be equivalent to the first result. This has been proven that the result of (i) and total result of (ii) are equivalent. We believe the integrated data in the proposed integrated object relational Web data warehouse system are successfully accessed and correctly integrated.

**CONCLUSION AND FUTURE WORK**

Our proposed Web integration data warehouse has utilized an object-oriented design concept to allow user requirements and underlying data source structures to be specified. It has successfully utilized a wider range of object-oriented features to model a simple integrated model that has a star schema structure in hierarchies. The hierarchies aim to allow the easiest information browsing of the data detail at the lowest level possible but at the same time fully capturing and representing data semantics in the underlying data source.

The complete range of data source models from different database design approaches, including well-structured data and semi-structured data, have been successfully integrated in the proposed model. Most importantly, large amounts of Web data such as HTML/XML are now formally handled in the proposed integrated data warehouse.

We have also implemented a prototype to allow an evaluation task performed on the integrated data and proposed technique. The correctness of the proposed integrated data warehouse model has been proved as well as the efficiency of performance among the queries has also been confirmed.

For future work, an immediate task could be to investigate the time performance of the queries. We would see an opportunity of using TKPROF trace tool to collect the statistics to analyze the processing time, fetching, and parsing of query.

**REFERENCES**


Breitbart, Y., Olson, Y., & Thompson, G. (1986). Database integration in a distributed heterogeneous data system. *ACM Data Engineering* (pp. 301-310).


Web Data Warehousing Convergence: From Schematic to Systematic


Web Data Warehousing Convergence: From Schematic to Systematic

International Journal of Data Warehousing and Mining, 1(3), 63-87.


This work was previously published in Int. Journal of Information Technology and Web Engineering, Vol 1, Issue 4, edited by G. Alkhatib and D. Rine, pp. 68-80, copyright 2006 by IGI Publishing (an imprint of IGI Global).