A Framework for Warehousing the Web Contents

Yan Zhu

DVS1, Department of Computer Science, Darmstadt University of Technology
D-64283 Darmstadt, Germany
Zhu@dvs1.informatik.tu-darmstadt.de

Abstract. This paper presents a framework for warehousing selected Web contents. In this framework, a hybrid (partially materialized) approach and extended ontologies are used to achieve Web data integration. This hybrid approach makes it possible to integrate DW data with Web-based information resources as they are needed. The Ontologies are used to represent domain knowledge related to Web sources and the logic model of data warehouses. Moreover, we define the mapping rules between Web data and attributes of data warehouses in the ontologies to facilitate the construction and maintenance requirements of data warehouses.

1 Introduction

Building data warehouses (DW) and efficiently implementing decision support or OLAP on them have become important tools for knowledge management and business intelligence. External data, such as, new product announcements from competitors, weather conditions at business activity location, and currency exchange rates may be needed to carry out meaningful OLAP. Ignoring this influence may cause the results to be incomplete, inexact or even totally incorrect [12, 15]. As Web technology develops, a vast amount of external data is already offered on the Web. If OLAP can take advantage of them, the resulting functions will be more powerful.

But warehousing Web data is a huge challenge. Some reasons are:

In a data warehousing environment, data is materialized in a DW and all queries are applied to the warehouse data to achieve a good query performance. Data in the DW is historical and nonvolatile. But most Web data is updated frequently, and the maintenance of a data warehouse that depends on Web data is more difficult compared with conventional data warehouses based on company-internal data. In contrast to the data warehousing approach, some Web data integration systems [1, 5, 6, 8, 10, 11, 19] adopt a virtual approach. In this approach, data remains at Web sources. When a query is posed, related data is integrated on the fly to give a reply. One of advantages is that data is not replicated, thereby guaranteeing to be consistent at query time. On the down side, the quality of Web data may not be as tightly controlled as that of the DW data. Moreover, since Web sources are autonomous, more sophisticated query optimization and execution methods are needed.

In order to reduce query response time, an appropriate set of Web data must be selected for materialization. On the other hand, some queries might not be answered only with materialized data in the DW. In order to support wider analyses on the DW, a virtual approach can be adopted to integrate needed Web data for producing a reply.

To integrate dynamically Web data with DW data, additional challenges must be overcome:

- The paradigm of the Web is radically different from the paradigm of the data warehouse. Data on the Web is mostly unstructured or semistructured, while a data warehouse is in most cases based on structured data in the relational or multidimensional data models.
- The process of integrating external (e.g. Web) data and internal warehouse data exacerbates a number of problems, such as, data format inconsistency, and semantic inconsistency.

In order to tackle these difficult problems, we have made the following efforts in our framework:

- Adoption of a hybrid approach. We select a set of nonvolatile or frequently queried Web data to materialize in the DW. With the aid of the virtual approach, some queries can be answered using Web data that is not materialized in the DW, or using a combination of warehouse data and Web data.
- Selection of the MIX model (Metadata based Integration model for data X-change) [3,4] as an internal data representation. This model represents data together with a description of their underlying interpretation context, and uses domain-specific ontologies to enable a semantically correct interpretation of the available data and metadata. Ontologies as a common interpretation basis can help to resolve the semantic inconsistency problem.
- Definition of mapping rules in a system-specific ontology to generate transformation codes semiautomatically. If Web data changes, e.g. because of the introduction of the Euro among the currencies, the ontology can be extended, the transformation rules can be modified, and mapping codes can be generated again, which relieves the maintenance task of data warehouses.

2 Related Efforts

In the Squirrel project [16] and the Lore system [20], the hybrid approach has already been discussed. The Squirrel project concentrates only on relational or object-oriented data sources. Its goal of using a hybrid approach is to develop a general and flexible mediator framework. Lore is a DBMS for managing semistructured information. Lore’s external data manager provides a mechanism for dynamically fetching and caching external data, and integrating them with data resident in the Lore system to answer a user’s query. The integrated external data is up-to-date. There are some important differences between our work and theirs. First, we integrate Web data in a DW, rather than an object repository. Compared with an object repository, a data warehouse based on a relational data model will introduce many different issues on the Web data transformation, the query performance requirement, or the warehouse maintenance. Second, in our framework, the partially materialized approach helps to answer queries that could not be answered based on DW data alone, and the dynamically integrated Web data is still historical and nonvolatile.

Among the systems for information integration that use an ontology, the DataFoundry project [9, 10] is similar to our research. It is based on a mediated data warehouse architecture with an ontology infrastructure, which is used to reduce the maintenance requirements of a warehouse. But some remarkable differences exist between our work and theirs. First, we focus on the Web data. Compared with data in scientific databases, Web data is unstructured or semistructured and has no explicit schema. Second, DataFoundry caches the most frequently accessed data to reduce query response time. In our framework, nonvolatile and frequently queried Web data is materialized in a DW.

The DWQ project [7, 8] also integrates information sources in the data warehousing environment. They consider two critical factors for the design and maintenance of particular data warehouse applications, which are conceptual modeling of the domain and reasoning support over the conceptual representation. This approach provides a system independent specification of the relationships between sources and between a source and an enterprise model at the conceptual level. By using an explicit conceptual model, the design phase and the maintenance phase of the information system can be facilitated. However the DWQ project focuses only on those sources that possess a schema. In our framework an extended ontology is used at the conceptual level, and also a metadata based object-oriented data model is used for the logical level. This makes it possible to integrate the Web data correctly and to generate transformation processes semiautomatically.

The rest of the paper is organized as follows. Section 3 gives a motivating example. The framework of the system is introduced briefly in Section 4. The ontology and a mediated data model are discussed in Section 5. Section 6 analyzes mapping mechanisms from the Web data to a star schema. The process of using an ontology to deduce transformation processes is discussed in Section 7. Section 8 will give the conclusion and discuss our future work.

3 A Tour Offer Data Warehouse Example

We define our example in a tour offer application. To simplify the problem description, we consider only tour packets for international city visits when the travel medium is aircraft. Using a tour offer data warehouse, the following inquiries are often analyzed, such as: Which city is the favorite choice in which month? What kind of hotel is booked most? What kind of influence does the local weather or currency exchange rates have on tour offers? Among inquired data, the weather data and the currency exchange rates probably lie outside the business data of a travel agency, but can be easily acquired from the Web.

In our framework, we select the past average weather data and the most common currency exchange rates for materialization in a DW. Fig.1 is the example of average weather data of Paris in 1998.

Sometimes, we also need to analyze, for example, why the monthly sold amounts of a tour offer to Spain dropped from September 1998 to January 1999. Perhaps the Spanish Peseta during that time became expensive. If we have no Peseta exchange rates in the DW, we could not answer such a query. Fortunately, the Federal Reserve Bank of St. Louis already offers historic monthly average exchange rates on the Web. We can use a virtual approach to integrate them, and then to recompose them with aggregation tour offer data from the DW to give the user a reply. Fig. 2 shows monthly currency exchange rate of Spanish Pesetas to one US$ on the Web.

A star schema of the tour offer data warehouse and a weather dimension table are defined in Fig.3.

01/06/99-Updated 06:19 PM ET

Paris, France, 1998						Temps °F
Month	Average H igh	Averag Low	Warmest Temp	Coldest Temp	Average dew point	Wet days
Jan	42	34	57	1	33	20
Feb	44	34	59	14	32	17
Mar	51	38	75	21	37	19
April	57	41	79	26	39	16
...						

Note: *Dew point is a humidity measure in Fahrenheit degrees;*
Wet days are days with rain or melted snow or ice totaling 0.01 inches

Fig. 1. Paris' average weather data in 1998

Date	EXSPUS
...	...
1998.08	151.72
1998.09	144.33
1998.10	139.23
1998.11	143.05
1998.12	142.08
1999.01	143.55
1999.02	148.52
...	

Average of daily figures, Noon buying rates in New York.
Source: Federal Reserve Bank of St. Louis and Federal Reserve Board of Governors

Fig. 2. The exchange rates of Spanish Pesetas to one US$

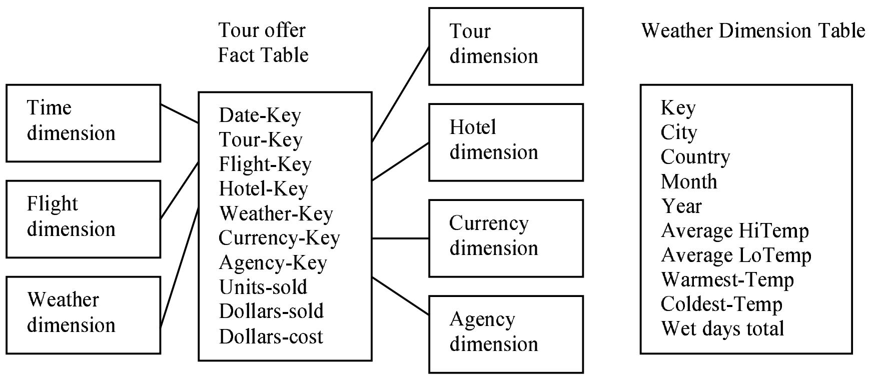

Fig. 3. The tour offer data warehouse

4 The Framework for Warehousing Web Data

Our framework showed in Fig.4 is based on a federation manager and a set of wrappers, which are introduced in [5]. In the framework, the federation manager (FM) uses a global mediated data model (MIX model) to represent data from Web sources.

The transformation processor (TP) transforms the data in the MIX model to the data in the DW, and loads the data into the DW. The task of the Incremental Maintenance Processor (IMP) is to calculate the incremental parts when Web data is updated. At first, the FM will integrate the updated Web data, the IMP will compare new MIX objects with the old copies and calculate the incremental parts. The incremental parts will also be transformed and loaded in the DW. The Query Processor (QP) provides the interface for querying the data warehouse. Upon receiving a query, the QP first determines the query on materialized data in the DW or on virtual data from the Web. If the query needs virtual data, the QP will decompose original queries, rewrite queries related to virtual data, and send them to the FM. After the replies are obtained from the Web and the DW, the QP recomposes them and sends the final answer to the user. The above process will be transparent to the user. The ontology repository consists of a domain-specific ontology, which conceptualizes domain knowledge related to the selected Web sources, and system-specific ontologies, which describe the data warehouse logical schema and define the relationship between the mediated model and the star schema of the DW. The Ontology Engine (OE) is an ontology processor, whereat concepts in the ontology can be implemented as classes, and relationships or functions in the ontology can be implemented as methods.

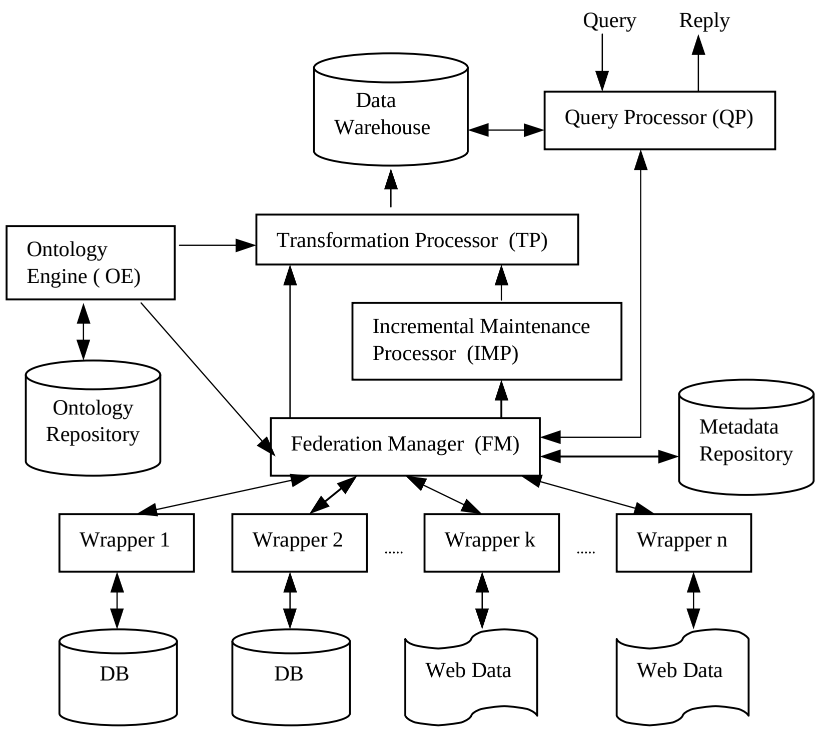

Fig. 4. The structure of warehousing Web data

```
Weather Ontology

(define-class CityName ( ?cityname)
        " A cityname is the name of a city."
        :def ( and (format ?cityname FullCityName)
                   (String  ?cityname)))
 (define-class YearString ( ?yearstring)
        "Yearstring is a kind of date notation, and is expressed as String."
        :def ( and (format ?yearstring "YYYY")
                   (string ?yearstring)))
(define-class YearNumber ( ?yearnumber)
        "Yearnumber is a kind of date notation according to ISO 8601specific year format, and is
        expressed as the number of years A. D."
        :def ( and (format ?yearnumber  YYYY)
                    (integer-range  ?yearnumber)
                         (= (i-lower-bound ?yearnumber) 0)
                         (= (i-upper-bound ?yearnumber) ∞)))
(define-class  AverageHighTemperature ( ?averagehigh)
        "An averagehigh is monthly average high temperature related to a city in a country. "
        :def (and (subclass-of  AverageHighTemperature  Weather-Temperature)
                   (or(unit ?averagehigh Fahrenheit)
                      (unit ?averagehigh Celsius ))
                      (float ?averagehigh )))
(define-function Fahrenheit-to-Celsius (?T_F) : -> ?T_C
        " A fahrenheit-to-celsius convert a temperature value in fahrenheit unit to the temperature
value in celsius unit, according to equivalence T_C =( T_F -32)*5/9. "
        :def (and (member ?T_F  Weather-Temperature )
                   (unit ?T_F  Fahrenheit)
                   (member ?T_C Weather-Temperature)
                   (unit ?T_C  Celsius)
                   (= (/ (* (- T_F 32) 5) 9) T_C )))
…
```

Fig. 5. The weather ontology

```
CompSemObj=
 <Weather, {
    <CityName, "Paris",            {< CityNameCode, "FullCityName">}>,
    <CountryName, "France",        {< CountryNameCode, "FullCountryName">}>,
    <Year, 1998,                   {< YearFormat, "YYYY">}>,
    <MonthlyAverageData,{
        <Month, 01,                {< MonthFormat, MM>}>,
        <AverageHigh, 42,          {< AverageHighTempByMonth,  "Fahrenheit">,<Scale, 1>}>,
        <AverageLow, 34,           {< AverageLowTempByMonth,  "Fahrenheit">,<Scale, 1>}>,
        <Warmest, 57,              {<WarmestTempInMonth,  "Fahrenheit">,<Scale, 1>}>,
        <Coldest, 1,               {<ColdestTempInMonth,  "Fahrenheit">,<Scale, 1>}>,
        <AverageDewPoint, 33,      {<AverageHumidityByMonth, "Fahrenheit">,<Scale, 1>}>,
        <WetDay, 20                {<TotalWetDayInMonth, "Day">,<Scale, 1>}>}>
    <MonthlyAverageData,{
        <Month, 02,                {< MonthFormat, MM>}>,
        <AverageHigh, 44,          {< AverageHighTempByMonth, "Fahrenheit">,<Scale, 1>}>,
         ...  }>
  ...  }>
```

Fig. 6. The integrated weather data represented based on MIX

5 The Ontology and the Mediated Data Model

Web data comes from various knowledge domains. These Web sources are developed independently and work autonomously. When we integrate Web data into a data warehouse, the inconsistencies exist not only between internal business data and external data on the Web, but also among Web data. Among these inconsistencies, semantically correct understanding and interpretation of related metadata is a key issue. A common ontology [13] serving as an agreement to use the shared knowledge in a coherent and consistent manner can help to tackle this problem. Fig. 5 is an example of the weather domain ontology, which is represented using Ontolingua [13], a knowledge representation language. Ontolingua definitions are Lisp-like forms that consist of a symbol, an argument list associated with the symbol, a documentation string, and a set of KIF [14] sentences labeled by keywords.

The MIX model is a self-describing object-oriented data model and based on the concept of a semantic object, which represents a data item together with its underlying semantic context. In MIX, in order to support the interpretation of the available data and metadata, each semantic object has a concept label associated with it that specifies the relationship between the object and the real world aspects it describes. These concept labels must be taken from a domain ontology. The MIX model may represent structured and semistructured data in a uniform way and on a common interpretation basis. In MIX, simple semantic objects (SemObj) represent atomic data items. Complex semantic objects (CompSemObj) represent objects consisting of multiple, possibly heterogeneous data elements, each of which describes exactly one attribute of the represented phenomenon.

An example of SemObj is:

< CityName, "Paris", {<CityNameCode,"FullCityName">}>

An example of CompSemObj is:

<MonthlyAverageData, {<Month, 01, {<Month Format, MM>}>, <...>,...}>

Fig.6 shows an example of the MIX model. More detailed definitions and explanations can be found in [3,4].

6 Mapping MIX Objects to the Attributes in the Star Schema

The MIX Model is an object-oriented data model, while the DW in our framework is based on the relational data model. Therefore, mapping MIX semantic objects to the data warehouse tables is roughly similar to mapping objects into relational tables [2, 17, 18, 21]. But a remarkable difference is that data in the DW is divided into facts in fact tables and descriptive attributes in dimension tables. Since business transaction data, which typically form the basis of the fact tables, will not be taken from the Web, we concentrate on mapping Web data to dimension tables. Between data values in MIX objects and attribute values in dimension tables, there could be one-to-one, one-to-many, many-to-many relationships. In the example of Section 3, the relationship between Web data values and attribute values in the dimension table is one-to-one. In some more complex cases, data values must at first be calculated, and then mapped to attribute values.

A MIX model consists of attributes and methods (conversion functions). Its attributes are either simple or complex objects. Therefore there exists a one-to-many composite aggregation relationship among parent object and children objects. In basic object-relational model mapping, we can map each complex object to a table, and implement composite aggregation in the parent table using foreign keys, which point to each child table. But in our example, weather dimensional table is redundant with city, country and year. Therefore, each complex object to a table can not be adopted in data warehousing. We map the root CompSemObj to one table in a way similar to One-Inheritance-Tree-One-Table approach. All equal ontology concepts are mapped to one column name, and all different concepts are mapped to separate column names. The physical representation of a value is mapped to an attribute value based on related column names. Semantic context associated with a semantic object can be separated from the database and stored to a metadata repository. These mapping processes can be done recursively.

The MIX model can also support inheritance, which is reflected in the inheritance of concepts in the domain ontology. Each MIX object only represents concrete integrated data values. Therefore, in the mapping we need not consider the inheritance relationship.

In the MIX model each object is identified through single or multipart key attributes, which is similar to the relational model. When an object is implemented in Java, a unique OID can be automatically assigned by the system. In a data warehousing environment, the primary key in a dimensional table can be an intelligent or non-intelligent key. In practice, unless we are absolutely certain that the identifiers will not be changed or reassigned in the future, it is safer to use non-intelligent keys. These non-intelligent keys can be automatically generated by the system. Therefore, when we map MIX objects to a dimensional table, we design an additional column in this table as primary key, one unique key value for each row can be generated by the system.

To map a nested complex semantic object structure to a table structure, we can first replace each CompSemObj with all its subobjects. This replacement will be processed recursively until there is no CompSemObj in the structure except the root CompSemObj. After the preprocessing, the mapping rules described as above are applied to this enlarged root complex semantic object.

When a semantic object occurs more than one time, a multiple attribute problem arises. For example, a book can have more than one author. But most relational databases do not support multi-valued attributes. To cope with this problem, we use separate rows to store each value of multi-valued attributes in dimensional tables.

7 Using Ontologies to Perform Transformations

In our framework, the ontology repository consists of the domain ontology and the specific ontologies about the description of the logical schema of the DW and mapping rules. The OE is an ontology processor, whereat concepts in the ontologies are implemented as classes, and relationships or functions in the ontologies are implemented as methods. The TP works under the guide of the schema of the DW and the mapping rules. Using ontologies, the transformation codes can be generated semi-automatically. As additional Web data is integrated, or when selected Web data is updated, the things we need to do are adding new concepts into the domain ontology, modifying mapping rules, then regenerating transformation codes.

Between data from Web sources and attributes in data warehouses, there are one-to-one, one-to-many, many-to-many relationships. These relationships also exist between the values in the MIX objects and the attributes in the data warehouse. In the mapping rules ontology, the correspondences are explicitly described, and the conversion functions are defined respectively. Fig. 7 is an example of the mapping rules ontology.

After the ontology has been defined, the OE can implement classes, relations and functions as Java classes and methods. By using mapping rules ontology, transformation methods can be deduced. The TP reads data represented in MIX objects and uses transformation methods to construct tables, generate column names and attribute values in the data warehouse. The process includes essential conversion of data type, data format, unit, etc.

```
 Mapping Rules Definition Ontology

(define-relation CityMapping (?city ?cityname)
"Mapping cityname to city. The cityname is a member of the CityName class in the Weather Ontology,  the
city is an attribute name of the WeatherSchema in the Weather Dimension Table Descriptions Ontology."
              :def ( and  (member ?city Weather-Dimension-Table-Descriptions-Ontology)
                          (member ?cityname Weather-Ontology)
                          (= (?city ?cityname))))

(define-relation YearMapping (?year ?yearstring)
"Mapping yearstring to year. The yearstring is a member of the YearString class in the Weather Ontology,
the year is an attribute name of the WeatherSchema in Weather Dimension Table Descriptions Ontology. "
              :def ( and  (member ?year Weather-Dimension-Table-Descriptions-Ontology)
                         (member ?yearstring Weather-Ontology)
                         (member ?yearnumber Weather-Ontology)
                         ((Year-Number-of  ?yearstring) ?yearnumber  ))
                         (= (?year ?yearnumber))))

(define-relation AverageHighMapping (?average-ht ?averagehigh)
"Mapping averagehigh to average-ht. The averagehigh is a member of the AverageHigh-Temperature class
in the Weather Ontology, the average-ht is an attribute name of the WeatherSchema in the Weather
Dimension Table Descriptions Ontology."
              :def ( and (member ?average-ht Weather-Dimension-Table-Descriptions-Ontology)
                         (member ?averagehigh Weather-Ontology)
                         (= (?average-ht ?Averagehigh))))
...
```

Fig. 7. Mapping rules definition ontology

8 Conclusions

In this paper we present a novel framework of integrating information contents from the Web for DSS and OLAP. Moreover, we also discuss how to use extended ontologies to transform the Web data for integration into data warehouses and to facilitate the maintenance requirements of the data warehouse.

At present, we are improving our framework and doing further research on the hybrid approach in a *Web Warehousing* environment. Some interesting issues we are currently looking into include how to select an appropriate set of Web data for materialization, the optimization problem of the Query Processor in the hybrid approach, the influence of Web data quality on a DW, and the further use of ontology and KR techniques in the design and maintenance of the data warehouses.

Acknowledgements

The author would like to thank Alejandro P. Buchmann for his detailed comments that helped improve this paper. The author would also like to thank Christof Bornhövd and Ronald Bourret for their productive discussions with regard to some parts of this paper.

References

1. Arens, Y., Knoblock, C. A., Shen, W-M.: *Query Reformulation for Dynamic Information Integration*, In Journal of Intelligent Information Systems, 6 (2/3) : 99-130, 1996
2. Blaha, M., Premerlani, W., Shen H.: *Converting OO Models into RDBMS Schema*, IEEE Software, 11(3) : 28-39, 1994
3. Bornhövd, C.: *MIX - A Representation Model for the Integration of Web-based Data*, Technical Report DVS98-1, Department of Computer Science, Darmstadt University of Technology, Nov., 1998
4. Bornhövd, C.: *Semantic Metadata for the Integration of Web-based Data for Electronic Commerce*, WECWIS'99, Santa Clara, USA,1999
5. Bornhövd, C., Buchmann, A. P.: *A Prototype for Metadata-based Integration of Internet Sources*, In Proc. of CAiSE'99, Heidelberg, Germany, June, 1999
6. Bayardo, R., Bohrer, W., Brice, R., et al: *InfoSleuth: Semantic Integration of Information in Open and Dynamic Environments*, In SIGMOD' 97, Tucson, Arizona, 1997
7. Calvanese, D., Giacomo, G. D., Lenzerini, M., and et al: *Description Logic Framework for Information Integration*, In Proc. of KR' 98, 1998
8. Calvanese, D., Giacomo, G. D., Lenzerini, M., et al: *Information Integration: Conceptual Modeling and Reasoning Support,* In Proc. of CoopIS'98, New York, 1998
9. Critchlow, T., Ganesh, M., Musick, R.: *Automatic Generation of Warehouse Mediators Using an Ontology Engine*, In Proc. of the 5th KRDB Workshop, Seattle, WA, 1998
10. Critchlow, T., Ganesh, M., Musick, R.: *Meta-Data based Mediator Generation*, In Proc. of CoopIS'98, New York, 1998
11. Chawathe, S., Garcia-Molina, H., Hammer, J., et al: *The TSIMMIS project: Integration of heterogeneous information sources*. In Proc. of IPSI'94, Japan, March, 1994
12. Damato, G. M.: *Strategic Information from External Sources*, White Paper, http://www.datawarehouse.com/, April, 1999,
13. Gruber, T. R.: *A Translation Approach to Portable Ontology Specifications,* Knowledge Acquisition, 5 (2): 199-220, 1993.
14. Genesereth, M.R., Fikes, R. E.: *Knowledge Interchange Format,* Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University
15. Hackathorn, R. D.: *Web Farming for the Data Warehouse*, Morgan Kaufmann Publishers, 1999
16. Hull, R., Zhou, G.: *A Framework for Supporting Data Integration Using the Materialized and Virtual Approaches,* In Proc. of ACM SIGMOD'96, Montreal, Canada, 1996
17. Keller, W.: *Mapping Objects to Tables*, In EuroPLoP' 97, 1997
18. Keller, W.: *Object/Relational Access Layers*, In EuroPLoP'98, 1998
19. Levy, A. Y., Rajaraman, A., Ordille, J. J.: *Querying Heterogeneous Information Sources Using Source Descriptions,* In Proc. of the 22nd VLDB Conference, Bombay, India, 1996
20. McHugh, J., Widom, J.: *Integrating Dynamically-Fetched External Information into a DBMS for Semistructured Data*, SIGMOD Record, 26 (4), 1997
21. Yoder, J.W., Johnson, R. E., Wilson, Q. D.: *Connecting Business Objects to Relational Databases*, In PLoP'98, 1998