Automated Data and Service Mapping for Integrated Electronic Markets

Stefan Böttcher , Sven Groppe , Tim Schattkowsky
University of Paderborn , Faculty 5 ,
Cooperative Computing & Communication Laboratory
Fürstenallee 11 , D-33102 Paderborn , Germany
email : stb@uni-paderborn.de , sg@uni-paderborn.de , tim.schattkowsky@c-lab.de

ABSTRACT

Currently there are multiple different catalog formats for product descriptions used in electronic markets and enterprise-internal applications. Whenever E-procurement applications rely on the catalog format, but want to integrate product descriptions that are stored in a different catalog format, a key problem is how to map different catalog formats onto each other. A further requirement is the integration of services (offered e.g. by remote market places) into enterprise applications that use a different catalog format. We demonstrate that mappings can be generated to combine services that operate on different catalog formats. Our approach avoids unnecessary manual work for mapping and automatically generates mappings between different services wherever possible. This allows us to run E-procurement applications on different catalogs with a fairly reduced manual work needed for mapping.

Keywords: electronic market place, classification, product catalog, data transformation, query reformulation, service mapping.

1. INTRODUCTION

Problem origin and motivation

E-procurement systems (e.g. electronic market places) handle their product data in XML formats like BMEcat [23], GOM [22], cXML [25], xCBL [24]. These XML formats provide for each product data a field for a code of a product group called category. Categories are arranged in a structure called classification[1] (c.f. figure 1).

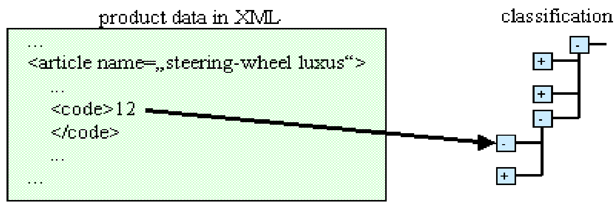

Figure 1: product data are handled in XML with reference to a position in a classification

A variety of standards have been suggested for classifications, e.g. ETIM [16], eCl@ss [19], Edibatec [18], UN/SPSC [17], NACE [21], PRODCOM [20], IEC 61360 [14], UniClass [15]. Additionally, many companies have their own classification for historic reasons. In order to interchange data between different E-procurement systems two tasks have to be solved: a conversion between the XML formats and a conversion of the product data according to the classifications are needed. While XML transformers (e.g. XSLT) usually handle the first task, our work in [7] contributes to save the second task, i.e. to *map* different classifications for the product data onto each other (c.f. figure 2).

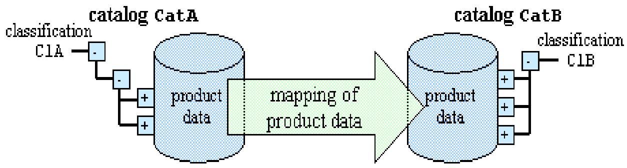

Figure 2: Data mapping from catalog `CatA` to catalog `CatB`

In E-procurement systems, client programs (e.g. `PA` and `PB`) are calling services (e.g. service `SerA` and service `SerB`) with parameters according to their used catalog that return results according to their used catalog (c.f. figure 3).

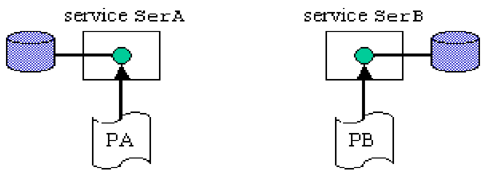

Figure 3: PA and PB call different services with data of different structure

A further step to integrate E-procurement systems is to apply *service mapping* (c.f. figure 4) to service calls of client programs (e.g. `PA`). Service mapping includes the mapping of service call, its parameters and result. The parameters (and result respectively) can be product data in the format of classification of service `SerA` (and service `SerB` respectively).

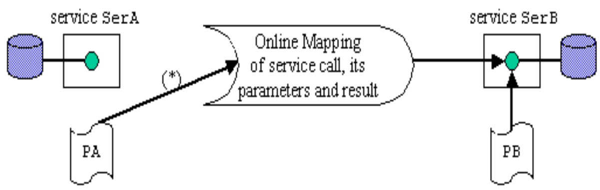

Figure 4: PA calls service `SerB`

Because mapping of services and classifications plays such a central role in integrating E-procurement systems and often the

[1] Usually a classification is defined hierarchically, but there is no need for defining the classification in such a way in the context of the paper. In figures here, however, classifications are presented hierarchically, nevertheless.

costs prohibits to develop a mapping manually, our goal is to minimize the manual work needed for mapping definitions by reversing and chaining already given mappings for product data, queries and service calls (c.f. figure 5). This contribution is a substantial profit for clients and suppliers participating in E-procurement systems, as clients of integrated E-procurement systems can access more offers of product than in a stand-alone E-procurement system and merchants in integrated E-Procurement systems can access a larger clientele than in a stand-alone E-procurement system. A client using service `SerA` accesses service `SerD` by applying a service mapping from `SerA` to `SerD` (c.f. figure 5). The service mapping from `SerA` to `SerD` is generated by reversing the given service mapping from `SerC` to `SerB` in order to get a service mapping from `SerB` to `SerC` (c.f. figure 5), then chain this generated service mapping from `SerB` to `SerC` and the given service mappings from `SerA` to `SerB` to a service mapping from `SerA` to `SerC`, at last chain this service mapping from `SerA` to `SerC` and the given service mapping from `SerC` to `SerD` to the necessary service mapping `SerA` to `SerD`.

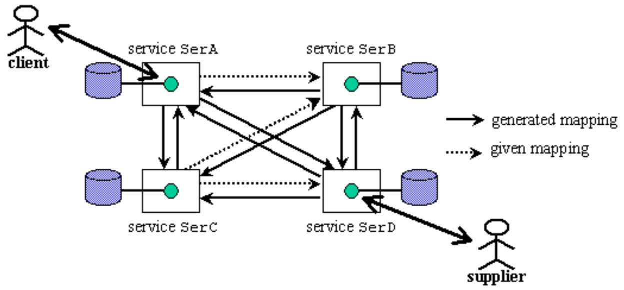

Figure 5: given and generated mappings

Our research origins from an integrated electronic market place developed at INCONY AG in Paderborn as part of the B2BECOM project of the European Community, that supports free exchange of data and services across the boundaries of different classifications.

Relation to other work and our focus

Mapping is related to contributions in database schema integration, views and query processing in federated databases, data warehouses and XML documents.

For the mapping of XML queries to other data storage formats at least two major research directions can be distinguished: first, to map XML queries to object oriented or relational databases (e.g. [29]) and second to map XML queries or documents to other XML documents (e.g. [6]). We follow the second approach, however additionally focus on mappings that overcome the problems of mismatching domains for services.

Within related contributions in schema integration two approaches to data and query reformulation can be distinguished. While the majority of contributions (e.g. [4],[5],[30]) map the data to a unique representation, we follow [2] and [3] to map the queries to those domains (or classifications) where the data resides. Similar to further work on query reformulation for federated databases [8] and data warehouses [10], [11], we use mappings that map schema information from one product catalog to another.

In contrast to all these contributions, we use such a mapping not only for data exchange but also for service exchange and for the automated generation of further mappings, i.e. reversed mappings and chained mappings for services.

Reversion of mappings has been discussed in the context of updating XML [27],[28]. Common to the database-XML mapping approaches, we also support data transfer in both directions, however we use a direct mapping from XML data to XML data, based on classifications that are used in catalog systems, without the need to map the data to a database, and we extend the mapping to service mapping. Another system containing a component for mapping of XML data between different XML structures is the BizTalk Mapper [26], a part of the BizTalk Server. As BizTalk, we support flexible and complex functions within the mapping definitions, but additionally we consider classifications and reverse mapping definitions. A further contribution [12] reverses rules of logic programming languages. However, this is done in the context of query reformulation and not in order to reverse a mapping definition. Additionally, we examine how to handle functions within the rules, how to add chaining of service mapping, and how to reverse a mapping definition. Finally, service integration and service mapping in distributed electronic market places have been considered e.g. in the context of agent technology (e.g. [1]). In contrast to these approaches, we use data mapping as basis for automated generation of reversed data mapping and service mapping.

We focused in [7] on the automated generation of mappings to integrate different catalog data formats into enterprise applications that require another catalog format and use this as basis for a free exchange of catalog data and queries in a distributed electronic market place.

Within this paper, we extend the approach in [7] to the automated generation of chained and reversed service mappings between different electronic market places, where a set of service mappings is given.

2. SERVICE MAPPING

In the following subsections, we describe how to define mapping for service calls and how a set of given mappings can be used in order to generate new mappings for service calls.

Defining service mapping

Whenever a market place `MB` offers a service that an application of a market place `MA` likes to use, then mapping the service (i.e. implement the service on `MA` in terms of the service on `MB`) may be the preferred alternative to copying (and mapping) all relevant product data of `MB` to market place `MA`. For the following examples, we assume that classification `ClA` has an additional product category `vw`, representing cars of the manufacturer `"VW"`, and classification `ClB` has a similar product category `car`, representing cars of any manufacturer. In the following example, a service on market place `MA` offers to install a `vw_steering-wheel SW` in a car of manufacturer `vw` called `VWCar`, and returns a modified `vw` car called `VWCarNew`, i.e.

```
installVW(vw VWCar, vw_steering-wheel SW,
          vw VWCarNew)
```

where `VWCar` and `SW` are input parameters and `VWCarNew` is an output parameter, which represents the car with the installed steering-wheel after the service call is completed. In general, we assume each service call to be implemented by a procedure call

$$s(is_1, \ldots, is_a,\ os_1, \ldots, os_b)$$

where $is_1, \ldots, is_a$ are the input parameters and $os_1, \ldots, os_b$ are the output parameters of the service.

Assume further, that market place MB offers a service comparable to installVW

```
install(car_manufacturer Manuf,
        car_wheel Wheel, car Car,
        car CarNew),
```

which installs a wheel Wheel in a car Car of manufacturer Manuf[2], where Manuf, Wheel, and Car are input parameters and CarNew is an output parameter, and car_manufacturer, car_wheel, and car are product categories of ClB.

The goal of service mapping is to implement the service (or part of the service) offered in market place MA by the service offered in market place MB. The problem to be solved is that MA and MB use different classifications, i.e. that although the service offered by MB may fulfill all the requirements needed for the service offered by MA, we still have to map the parameters of the service call in order to use the service of MB to contribute to our service on MA.

Given a data mapping M from ClA to ClB as declared in [7] and a reverse mapping M^{-1} from ClB to ClA, which can be generated from M by the approach presented in [7], map is the function, that applies mapping M to product data of ClA in order to generate a representation of these product data according to ClB.

For example, map applied with mapping M to product data SW of the product category vw_steering-wheel of catalog CatA generates an instance Wheel of the product category car_wheel of ClB. This is written as

```
set Wheel=map(SW,M⁻¹).
```

Analogously, map applied with mapping M to an instance VWCar of category vw generates an instance Car of category car defined in ClB . This is written as

```
set Car=map(VWCar,M⁻¹).
```

Finally, map applied with the reverse mapping M^{-1} to an object CarNew of category car of ClB generates an instance VWCarNew of the category vw defined in ClA . This is written as

```
set VWCarNew=map(CarNew,M⁻¹).
```

Then, a service call to

```
installVW(VWCar, SW, VWCarNew)
```

on market place MA can be mapped to a service call

```
install(Manufacturer, Wheel, Car, CarNew)
```

on market place MB by following rule (with install and installVW being defined as before) :

Example 1:

```
installVW(VWCar, SW, VWCarNew):-
  set Manufacturer="VW",
  set Wheel=map(SW,M),
  set Car=map(VWCar,M),
  install(Manufacturer,Wheel,Car,CarNew),
  set VWCarNew=map(CarNew,M⁻¹).
```

[2] Manuf must be one of "VW" and "Opel".

In general, in order to implement a service s on one market place by a service t[3] on another market place, we need a mapping that defines how input and output parameters are mapped to each other. The mapping rule is:

```
s(is₁,…,isₐ, os₁,…,os_b):-
    set it₁=f_i1(i_k1,1,…,i_k1,e1),
    …,
    set it_c=f_ic(i_kc,1,…,i_kc,ec),
    check condition,
    t(it₁,…,it_c, ot₁,…,ot_d),
    set os₁=f_o1(o_m1,1,…,o_m1,g1),
    …,
    set os_b=f_ob(o_mb,1,…,o_mb,gb).
```

where $is_1,\ldots,is_a$ are input parameters and $os_1,\ldots,os_b$ are output parameters of the service s to be implemented, and $it_1,\ldots,it_c$ are input parameters and $ot_1,\ldots,ot_d$ are output parameters of the service t that is used to implement s, and each $i_{ki,j}$ is one of the input parameters $is_1,\ldots,is_a$ and each $o_{mi,j}$ is one of the output parameters $ot_1,\ldots,ot_d$. Furthermore, condition restricts the input parameters $it_1,\ldots,it_c$ of the service t.

In general, a *condition* condition of the mapping rule is defined as follows. true and false are conditions. If Attr is an attribute defined for a category source, const is a constant value, and op is a comparison operator (<,>,=,≠,≤,≥), then source.Attr op const is a condition. If B1 and B2 are conditions, then B1 and B2, B1 or B2 and not B1 are conditions too.

The assignments to $it_1,\ldots,it_c$ have to be mapped before the service t is called, and the output parameters $os_1,\ldots,os_b$ must be mapped after the call of t. Whenever a service has a parameter used for input and output, the rule above can be applied by listing such a parameter in both lists of input parameters and output parameters.

Chained service mapping

As with data mapping, it is possible to chain service mapping, i.e. if market place MA implements a service s that is mapped to a service t of market place MB, MB may map this service t to a service u of market place MC on the basis of a given data mapping M between the classifications used in MB and MC. Instead of executing the service t of market place MB, we can chain the service mapping definitions (see Figure 6) such that market place MA can call directly the service u of market place MC. The function query_map($condition_t$,M) reformulates the query $condition_t$ according to the data mapping M as described in [7].

Reversed service mapping

Assume, we have already a service mapping that implements a service, say installVW, on a market place MA by a service, say Install, on market place MB. With this service mapping, we can use product data of a catalog CatB in MB in the service offered on market place MA.

When we now also want to compute a service mapping in the opposite direction, e.g. in order to use the product data of a catalog CatA in MA and the service installVW on market place MA in an application running on market place MB, then there are different possibilities how to compute such a *reversed service mapping* from installVW to install.

[3] with the same functionality

(1) $s(is_1,\dots,is_a,\ os_1,\dots,os_b)$:-
    set $it_1=f_{it1}(is_{k1,1},\dots,is_{k1,e1})$,
    …,
    set $it_c=f_{itc}(is_{kc,1},\dots,is_{kc,ec})$,
    check $condition_t$,
    $t(it_1,\dots,it_c,\ ot_1,\dots,ot_d)$,
    set $os_1=f_{os1}(ot_{m1,1},\dots,ot_{m1,g1})$,
    …,
    set $os_b=f_{osb}(ot_{mb,1},\dots,ot_{mb,gb})$.

(2) $t(it_1,\dots,it_k,\ ot_1,\dots,ot_d)$:-
    set $iu_1=f_{iu1}(it_{n1,1},\dots,it_{n1,p1})$,
    …,
    set $iu_j=f_{iuj}(it_{nj,1},\dots,it_{nj,pj})$,
    check $condition_u$,
    $u(iu_1,\dots,iu_j,\ ou_1,\dots,ou_h)$,
    set $ot_1=f_{ot1}(ou_{q1,1},\dots,ou_{q1,r1})$,
    …,
    set $ot_h=f_{oth}(ou_{qh,1},\dots,ou_{qh,rh})$.

(3) $s(is_1,\dots,is_a,\ os_1,\dots,os_b)$:-
    set $iu_1=f_{iu1}(f_{it_{n1,1}}(is_{k_{n1,1},1},\dots,is_{k_{n1,1},e_{n1,1}}),\dots)$,
    …,
    set $iu_j=\dots$,
    check query_map($condition_t$,M) and $condition_u$,
    $u(iu_1,\dots,iu_j,\ ou_1,\dots,ou_h)$,
    set $os_1=f_{os1}(f_{ot_{m1,1}}(ou_{q_{m1,1},1},\dots,ou_{q_{m1,1},r_{m1,1}}),\dots)$,
    …,
    set $os_b=\dots$.

Figure 6: Chaining of service mapping definitions

$s(is_1,\dots,is_a,\ os_1,\dots,os_b)$:-
    set $it_1=f_{it1}(is_{k1,1},\dots,is_{k1,e1})$,
    …,
    set $it_c=f_{itc}(is_{kc,1},\dots,is_{kc,ec})$,
    check $condition_t$,
    $t(it_1,\dots,it_c,\ ot_1,\dots,ot_d)$,
    set $os_1=f_{os1}(ot_{m1,1},\dots,ot_{m1,g1})$,
    …,
    set $os_b=f_{osb}(ot_{mb,1},\dots,ot_{mb,gb})$.

$t(it_1,\dots,it_c,\ ot_1,\dots,ot_d)$:-
    set $is_1=f_{it_{p1}}{}^{-1}(it_{p1})|_{is_1}$,
    …,
    set $is_a=f_{it_{pa}}{}^{-1}(it_{pa})|_{is_a}$,
    check query_map($condition_t$,$M_{st}{}^{-1}$),
    $s(is_1,\dots,is_a,\ os_1,\dots,os_b)$,
    set $ot_1=f_{os_{q1}}{}^{-1}(os_{q1})|_{ot_1}$,
    …,
    set $ot_d=f_{os_{qd}}{}^{-1}(os_{qd})|_{ot_d}$.

Figure 7: Reversing a service mapping definition

One possibility is to use the data mappings M and M^{-1} as before to compute the new service mapping by just exchanging the role of CatA and CatB. The disadvantage of this possibility is that it is necessary to identify corresponding parameters of the services (and to map them by reverse functions) a second time.

In order to avoid this, it is alternatively possible to take the given service mapping as input and to generate a reversed service mapping, similar to as we generate a reversed data mapping.

The reversed service mapping of the service mapping shown in Example 1 is

Example 2:

```
install(Manufacturer,Wheel,Car,CarNew):-
    check Manufacturer="VW",
    set SW=map(Wheel,M^-1),
    set VWCar=map(Car,M^-1),
    installVW(VWCar,SW,VWCarNew),
    set CarNew=map(VWCarNew,M).
```

This reversed mapping can be used to implement a service on market place MB (install) by calling a service of market place MA (installVW).

The general approach to reverse a service mapping is shown in Figure 7. M_{st} is the data mapping from the classification used in service s to the classification used in service t. The reversed mapping $M_{st}{}^{-1}$ can be computed by the approach described in [7]. The used functions f_x must be invertible and in order to retrieve the needed parameters is_1, …, is_a (and ot_1, …, ot_d respectively), we must choose the correct p1, …, pa in {1,…,c} (and the correct q1, …, qd in {1,…,b} respectively). The function query_map($condition_t$,$M_{st}{}^{-1}$) reformulates the condition $condition_t$ given as parameter according to the data mapping $M_{st}{}^{-1}$ given as parameter as described in [7].

3. SUMMARY AND CONCLUSIONS

Product catalogs of engineering companies with a wide product portfolio store products according to classifications, which are also used by enterprise applications that access such a catalog. However, these catalogues are not just placed as an island as online catalogue on the companies website but is also integrated into customers' E-procurement systems or marketplaces, together with the products of other manufacturers. Those company-spanning systems, in general, use different classifications, in most cases less specific than those of the companies or with different structures. Therefore the catalog integration requires a mapping between the classifications. Since the manual map definition between different classifications requires considerable work and is error prone, we suggest instead generating mappings automatically wherever possible. This includes the automated generation of reversed mappings and chained mappings. We presented, how to generate these mappings for service mapping.
This allows a service which has been implemented for a single catalog format to use data of all other catalogs for which a mapping exists, and it allows to use services of other market places with the locally stored catalog data. We consider this exchange of data and services across the boundaries of company specific or market place specific catalog formats to be an important step towards enterprise application integration. Although we developed our approach in the context of electronic market places, it might be promising to investigate how this can be extended to other areas of enterprise application integration.

5. REFERENCE

[1] Alon Y. Levy, Anand Rajaraman, Joann J. Ordille: Query-Answering Algorithms for Information Agents. **AAAI/IAAI**, Vol. 1 1996.
[2] Chen-Chuan K. Chang, Hector Garcia-Molina: Approximate Query Translation Across Heterogeneous Information Sources. **VLDB 2000**.
[3] C.-C. K. Chang and H. Garcia-Molina. Mind your vocabulary: Query mapping across heterogeneous information sources. **In Proc. of the 1999 ACM SIGMOD Conf.**, Philadelphia, 1999. ACM Press, NY.
[4] S. Cluet, C. Delobel, J. Simon, and K. Smaga. Your mediators need data conversion! **In Proc. of the 1998 ACM SIGMOD Conf.**
[5] S. Abiteboul, S. Cluet, and T. Milo. Correspondence and translation for heterogeneous data. **In Proc. of the 6th ICDT**, 1997.
[6] Serge Abiteboul: On views and XML. **In PODS**, pages 1-9, 1999.
[7] Stefan Böttcher, and Sven Groppe: Automated Data Mapping for Cross Enterprise Data Integration, **International Conference of Enterprise Information Systems (ICEIS 2003)**, Angers, France, 2003.
[8] Jaques Calmet, Sebastian Jekutsch, and Joachim Schu: A generic query-translation framework for a mediator architecture. In **ICDE**, pages 434-443, 1997.
[9] Alon Y. Levy and Daniel S. Weld: Intelligent internet-systems. **Artificial Intelligence**, 118(1-2), 2000.
[10] L. Yan, R. J. Miller, L. M. Haas, and R. Fagin: Data-driven understanding and refinement of schema mappings. **In ACM SIGMOD, Int. Conf.**, Santa Barbara, 2001.
[11] Renée J. Miller, Laura M. Haas, and Mauricio A. Hernández: Schema mapping as query discovery. In VLDB 2000, **Proceedings of 26th International Conference on Very Large Data Bases**, September 10-14, 2000. Cairo, Egypt, pages 77-88, Morgan Kaufmann, 2000.
[12] Oliver M. Duschka, Michael R. Genesereth, and Alon Y. Levy: Recursive query plans for data integration. **Journal of Logic Programming**, 43(1):49-73, 2000.
[13] W3C: Extensible Stylesheet Language (XSL). **http://www.w3.org/Style/XSL/**, 2001.
[14] International Electrotechnical Commission (IEC): IEC 61360-1 to 4, Standard data element types with associated classification scheme for electric components. **http://www.iec.ch/**, 2002.
[15] International Organization for Standardization (ISO): Classification of information in the construction industry. **Technical Report ISO 14177**, Geneva: International Organization for Standardization. (E)1994.
[16] ETIM Deutschland: ETIM. **http://etim.de/**, 2002.
[17] Electronic Commerce Code Management Association (ECCMA): Universal Standard Products and Services Classification (UNSPSC). **http://www.unspsc.org/**, 2002.
[18] Edibatec: Bienvenue sur le site Edibatec. **http://www.edibatec.org/**, 2002.
[19] eCl@ss e.V.: eCl@ss - Standard für Materialklassen und Warengruppen. **http://www.eClass.de**, 2002. eCl@ss e.V. c/o Institut der deutschen Wirtschaft, Köln.
[20] Commission of the European Communities (Statistical Office/Eurostat): PRODCOM list (List of PRODucts of the European COMmunity), 2002 version. Luxembourg: Office for Official Publications of the European Communities, 2002.
[21] Commission of the European Communities (Statistical Office/Eurostat): General Industrial Classification of Economic Activities within the European Communities. Luxembourg: Office for Official Publications of the European Communities, (reprint of 1970 edition) 1985.
[22] CEN/ISSS Workshop on Global Engineering Networking: Definition of an information model to enable the interchange of engineering information on a marketplace and an appropriate description of content in a system-neutral format. **Draft**, 2000.
[23] Volker Schmitz, Oliver Kelkar, Thorsten Pastoors, Thomas Renner und Claus Hümpel: Spezifikation BMEcat Version 1.2. **http://www.bmecat.org/**, Fraunhofer IAO, Universität Esssen BLI, 2001.
[24] Commerce One, Inc.: XML Common Business Library. **http://www.xCBL.org/**, 2002.
[25] Ariba, Inc.: CXML User's Guide Version 1.2.007. **http://www.cXML.org/**, Ariba, Inc., 2001.
[26] Microsoft Corporation: Microsoft BizTalk Server 2002 Enterprise Edition. Documentation, **http://www.microsoft.com/biztalk/**, Microsoft Corporation, 2002.
[27] Tatarinov, I., Ives, Z.G., Halevy, A.Y., Weld, D.S.: Updating XML. **ACM SIGMOD Int. Conf. on Management of Data**, 2001.
[28] Zhang, X., Mitchell, G., Lee, W.C., Rundensteiner, E.A.: Clock: Synchronizing Internal Relational Storage with External XML Documents, **RIDE-DM** 2001.
[29] Bourret, R., Bornhövd, C., Buchmann, A.P.: A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases. **2nd Int. Workshop on Advanced Issues of EC and Web-based Information Systems (WECWIS)**, San Jose, California, June, 2000
[30] E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. **Trans. on Database Systems**, 19(2), 1994.