UNIVERSITY
OF TRENTO

**DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY**

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

# PRELIMINARY EVALUATION OF SCHEMA MATCHING SYSTEMS

Mikalai Yatskevich

November 2003

Technical Report # DIT-03-028

# Preliminary Evaluation of Schema Matching Systems

Mikalai Yatskevich [1,2]

[1] DIT – Dept. of Information and Communication Technologies, University of Trento, 38050 Povo, Trento, Italy,
[2] BSU – Belarusian State University 220000 Skorini Ave 4, Minsk, Belarus
yatskevi@dit.unitn.it

**Abstract.** This evaluation of the state-of-the-art schema matching approaches is based on a comprehensive testing of modern systems on various real-world schemas. The results obtained show that there is no matcher that performs best on all types of schemas used. The quality of mappings depends significantly on the approaches to schema matching used and on the tuning of the matcher to the particular schema type. The approach proposed in COMA that combines results of different schema matchers proved its effectiveness in our evaluation.

## 1. Introduction

Matching is the task of finding semantic correspondences between elements of two schemas [12, 13, 16, 10]. Some important application domains are data integration and coordination (Theoretical foundations of this issues can be found in [8, 9, 18]), data warehousing, semantic query processing [17, 6, 10] and web services coordination. Among recently developed schema matching systems only a few approaches (Clio [16], COMA [5], Cupid [14], and Similarity Flooding (SF) [15]) perform the generic schema matching which can be applied to different applications and schema languages.

According to [17, 10].schema matching approaches can be classified as follows:

- *Hybrid or composite*. Hybrid matchers use multiple matching criteria. A composite matcher combines results obtained by exploiting several matching algorithms.
- *Weak or strong semantics*. Weak semantics techniques do not use any semantic information (e.g. synonymy / hyponymy relationships) and represent output as a coefficient in the [0…1] range. Strong semantics techniques use semantic information and represent their output using semantic relations ($\equiv, \perp, \subseteq, \supseteq$).
- *Instance based or schema based*. Instance based matchers consider data instances. Schema based matchers rely only on schema level information.
- *Element or structure level*. At the element level, matching is performed on individual elements. Structure level matchers consider combinations of elements such as complex schema structures.
- *Language or constraint based*. Language based matchers exploit linguistic approaches i.e., comparing names of elements. Constraint based matchers exploit constraint information i.e., relations, keys.

In our evaluation, we use weak semantics schema based matchers representing both hybrid and composite approaches. They exploit various element and structure level techniques of analysis language, and constrained based information.

This work is strongly influenced by the survey in [4]. In contrast to [4], in our work we provide real time evaluations of matching prototypes, rather than surveying the results cited elsewhere. We propose time measures as an important and valuable part of schema matchers' evaluation. In contrast to the existing evaluations, we keep uniform conditions for all matchers and use the same test schemas for all matching prototypes in order to obtain comparable results.

The report is organized as follows. In Section 2, we discuss the comparison criteria we use. In Section 3, we introduce the systems being evaluated and present some information about their inner structure and approaches to schema matching. In Section 4, we review the results produced by the matching systems. In Section 5, we compare the results of matching systems by summarizing their strengths and weakness. Section 6 concludes the paper and discusses some future work.

## 2. Comparison criteria and methodology.

While comparing different schema matching systems we are mainly based on the framework presented in [4]. This work compares different schema matching evaluations based on criteria from the 4 main areas: *Input*, *Output*, *Quality measures*, and *Effort*.

We add the 5th set of criteria, *Time measures*. Time performance of schema matchers is often not very important however in some applications such as communication in peer-to-peer networks, the time needed to find "good enough" mapping plays the key role [3, 7].

### 2.1. Input: test problems

We consider two main categories of information about the test schemas.

The first is *Schema type and language*. There are many types of schemas such as relational, XML, RDF schemas, onthologies exists. Heterogeneity in structure, type and language of the schemas can reveal different facets of the match algorithms. Thus, some approaches can perform better or worse depending on the particular schema properties.

*Schema information* is the second main category of input information. Number of schema elements, number of mappings, their ratio and direction of matching can influence the speed and quality of matching significantly.

### 2.2. Output: match result

Currently, the output of most matching systems is a set of the correspondences between attributes of schemas. Each correspondence has a similarity value in [0..1]. Element representation can play very important role in finding and representing results of matching. A graph model is a typical internal representation of schemas in the matching systems. Thus, schema elements can be represented by either nodes or paths in the schema graphs.

Cardinality of matching is restricted to 1:1 local cardinality for most state-of-the-art systems. Thus, matchers try to find one match candidate from the target schema for each element of the source.

## 2.3. Match quality measures

In order to determine the quality of matching, the results obtained by automatic matcher (P) usually are compared with an oracle. The real match result (R) obtained by solving task manually can be used as such oracle. In our study, we based on such common measures of saving manual effort as *Precision, Recall, Overall,* and *F-measure* as defined in [4, 5].

Correctly determined matches are identified as *I*. False matches as *F=P\I*, and missed matches as *M=R\I*. Based on the cardinalities of these sets, the following quality measures are used in [1,2,5,15]:

$$Precision = \frac{[I]}{[P]} = \frac{[I]}{[I]+[F]}$$

$$Recall = \frac{[I]}{[R]}$$

$$F-Measure = 2*\frac{Precision * Recall}{Precision + Recall}$$

$$Overall = 1-\frac{[F]+[M]}{[R]} = \frac{[I]-[F]}{[R]} = Recall*\left(2-\frac{1}{Presision}\right)$$

*F-measure* represents the harmonic mean of *Precision* and *Recall*. *Overall* [15, 5] was developed specifically in the schema matching context. The main underlying idea of *Overall* is to quantify the post-match effort needed for adding missed matches and removing false ones.

## 2.4. Time measures

Time measures are often important, for instance when matchers work with no human interaction involved. This situation appears for instance in peer-to-peer networks [7]. Thus, matching result should not only be precise enough but also obtaining of this result should be fast enough for real-time computation.

In our study, we measured the time schema matchers need to produce a mapping compared with the number of schema elements and mappings to produce. The results obtained are treated qualitatively to reason about complexity of schema matching tasks for each matcher tested.

All tests were performed uniformly on a P4-2000 computer with 512 Megabytes of RAM under Windows 2000 operation system with no other applications running.

### 2.5. Test methodology

Quantifying the user efforts is one of the main requirements to the matcher prototypes while performing matching in semiautomatic way. Both the *pre-match efforts* and the *post-match efforts* should be taken into account.

According to [4] the pre-match efforts include:

- Training of the machine learning-based matchers;
- Configuration of the various parameters of the match algorithms, e.g., setting different threshold and weight values;
- Specification of an auxiliary information, such as, domain synonyms and constraints ;

There are many ways to quantify post-match effort [4]. 1-*Recall* and 1-*Precision* estimates the effort to add false negatives and to remove false positives respectively. *Overall* and *F-Measure* take into account both *Recall* and *Precision* in order to provide a comprehensive measure of the post-match user efforts.

## 3. Matching systems

COMA, Cupid, and SF are among the most well-known and recent schema matching systems. In the following, we shortly describe their key features, their approaches to matching and the techniques used.

### 3.1. COMA

COMA [5] is a composite matcher. COMA provides an extensible library of different matchers and supports various aggregating and selecting strategies. The matchers exploit both the structure and element level schema information. Special matchers reuse results from previous matches and user interaction. Schemas are transformed to inner representations (rooted directed acyclic graphs). Complete paths from the root of the schema graph to the corresponding node uniquely identify each schema element.

### 3.2. Cupid

Cupid [14] combines a name matcher with a structural match algorithm in a sophisticated hybrid manner. Name and data type similarity values are combined with structure level heuristic to provide better result. Schemas are converted into trees and additional nodes are added to resolve multiple relationships between a shared node and its parent nodes.

### 3.3. Similarity Flooding (SF)

SF [15] converts schemas (SQL DDL, RDF, XML) into its inner representation (labeled graphs) and uses fix-point computation to determine correspondences between nodes of the graphs. Results of a simple element level name string matcher are fed into structural based on fix-point computation one. SF does not exploit any external dictionary and provides various filters to select the "best" matches from results obtained by the structural matcher.

## 4. Evaluation

We have tested systems described above on four pairs of real world schemas, which can be found in [11]. There are six XML schemas and two SQL DDL schemas with number of elements ranging from 10 to 80. In our tests, we did not use any mapping and schema reuse techniques. (They were turned off whenever available in the prototypes). The same (with respect to difference in the dictionary file formats) synonym dictionary were used for COMA and Cupid. It was merged from default synonym files of the systems. Finally, we fed the schemas into the systems in both a forward and a backward way.

### 4.1. COMA

COMA with its best combination of matchers described in [5] (*NamePath+Leaves* matcher, *Delta* select strategy and *Average* aggregation strategy) showed the best results among all the tested matching prototypes. The average *Overall* value was 0.53. Maximum *Precision* was 0.94. Recall values are also relatively high. Other combinations of matchers and strategies showed worse results and are not considered further in this report. COMA showed its best on large schemas what can be explained by inner structure of the matchers used.

According to [5], *NamePath* matches elements based on their hierarchical names (concatenation of the names of all elements in the path from root to element). Then *Name* (the matcher which combines results of *Affix, Trigram*, and *Synonym* matchers) is applied to compute the similarity among the hierarchical names. Considering the complete name path of an element allows the exploitation of not only lexical but also structural information, which may improve match accuracy.

*Leaves* [5] takes into account only leaf elements (element level similarity is estimated using *NameType* matcher) to compute the similarity between two inner elements. This strategy, according to [5], performs better in the cases of structural conflicts.

Relatively poor results for small schemas can be explained considering lack of structural properties in the small schemas and their big influence (both matchers *NamePath* and *Leaves* use structural information in form of paths to element and leaves of element) on the results obtained.

Execution times of COMA and SF are shown in Figure 1.

COMA with the *NamePath+Leaves* matcher combination is the fastest prototype in our evaluation. The computation time depends mainly on number of mappings.

COMA is the only system in our evaluation whose results do not depend on the order in which schemas are compared. It can be explained by noticing that COMA (by default) performs matching in both directions. Then it selects feasible match pair exploiting information from both unidirectional mappings.

The pre-matching effort with COMA is minimal. COMA converts XML and relational schemas into an inner representation format (graphs) and performs matching. Matching results are represented as strings with the full path to identify elements of both schemas and value [0…1] representing similarity between them.

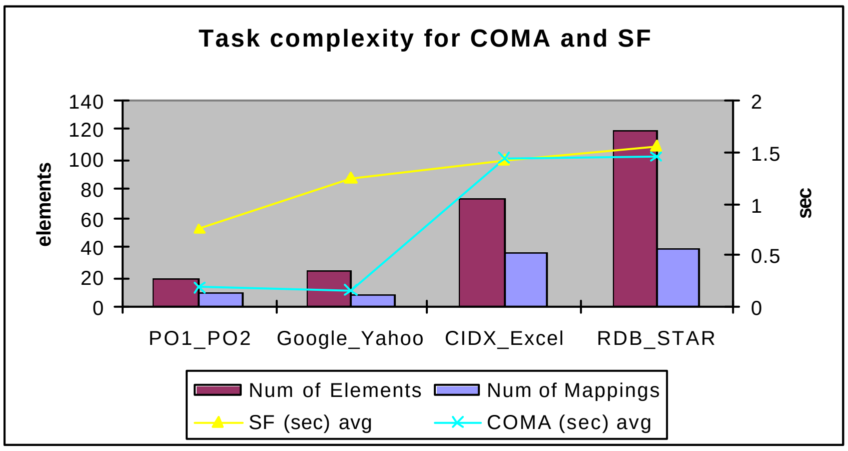


**Fig. 1.** Time SF and COMA needed to perform matching on different schemas and task complexity

A user can improve the quality of matching using the special matcher.

### 4.2. Cupid

Cupid showed the second highest average *Overall* and *F-Measure* in our evaluation. The sophisticated hybrid algorithm gives Cupid an advantage on small schemas. The best results (*Precision* 1.0, *Overall* 0.67) among all other matchers are obtained on PO-Simple schemas. The worst result (*Precision* 0.53, *Overall* 0.05) is obtained on the biggest schema RDB-STAR.

According to [14], Cupid considers elements with similar (within factor of 2) number of leaves in their subtrees. Thus, the poor results on some schemas (e.g. RDB-STAR) can be explained by noticing that Cupid can prune out relevant matches.

Cupid results depend on the order in which schemas are compared to each other. One possibility is because Cupid generates 1:n output, in order to conform to the input requirements of BizTalk Mapper. This is problem due the software implementation more than the algorithm.

Cupid can match schemas in XML Biztalk Mapper schema format. The results are written to an output file, which can be opened by Biztalk Mapper. User can improve the matching results by using its graphical user interface.

The results of Cupid can be adjusted by changing the threshold values; this allows the user to control the desired mapping quality.

### 4.3. Similarity Flooding (SF)

Similarity Flooding converts relational and XML schemas into an internal representation. We tested SF as part of the Rondo model management system. All results are represented through a useful graphical user interface. The main characteristic of SF is its purely structural approach. It does not use any synonym dictionary. Without dictionaries on our tests, the other matching systems perform worse than SF. SF performs better on “well-structured” schemas with more structure level information. It has the maximum average *Precision* and the minimum average *Recall* among systems tested. SF, like Cupid, does not produce the same results in forward and backward mapping.

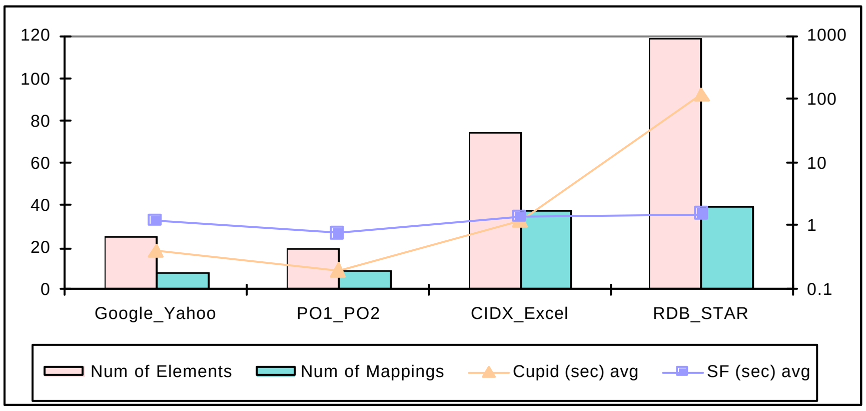


**Fig. 2.** Time SF and Cupid needed to perform matching on different schemas and task complexity

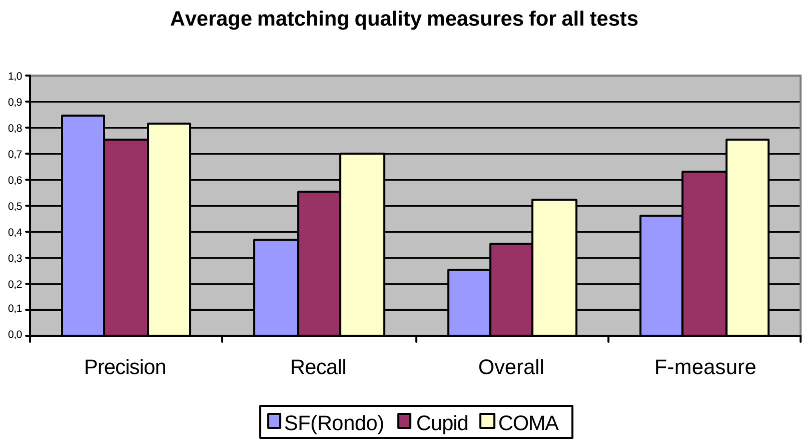


**Fig. 3.** Average matching quality values for all matchers

Figures 1 and 3 show that on large schemas SF is faster than Cupid and performs nearly as fast as COMA. SF is considerably slower then COMA and Cupid on small schemas.

## 5. Comparative evaluation

Average quality measures obtained are shown on the Figure 3. The results for each particular schema pair are depicted on the Figure 4.

The best average results are obtained by COMA and Cupid. COMA performs best on the large schemas. Cupid is the best matching system for small schemas. SF shows its structure nature and performs reasonably well on schemas with similar structures. It showed the maximum average *Precision* but minimum average *Recall* and *Overall* in our evaluation.

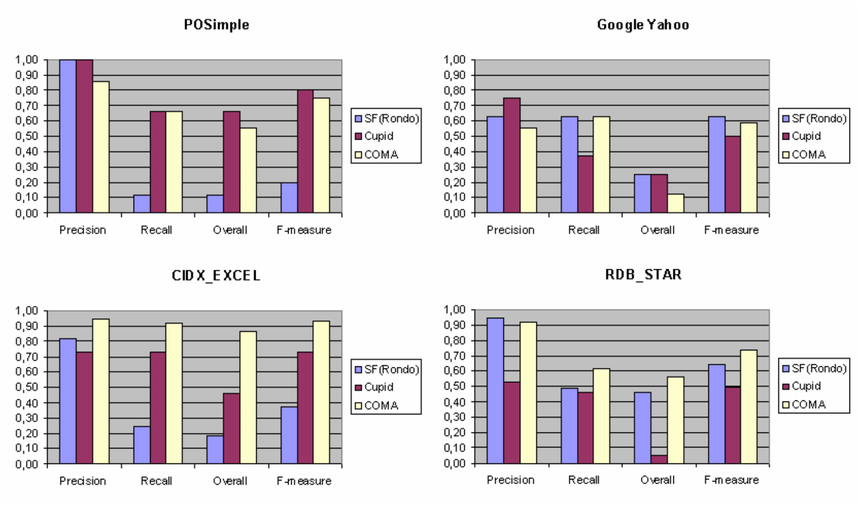


**Fig. 4.** Matching quality results obtained on 4 pairs of testing schemas.

## 6. Conclusion and future work

Despite the considerable progress on schema matching there is still no best matcher for schemas of all types and sizes. The results returned depend significantly on the approaches to schema matching used and on the tuning of the matcher to the particular schema type.

This work is very preliminary. Thus, this report can be extended in the following ways.

- *Number of schemas used*. Using more schemas of different types and sizes can help in collecting necessary information about strong and weak points of each matching system.
- *Number of matchers used*. Testing various matching prototypes can provide more information about approaches used in them. Especially interesting seems testing the systems based on Semantic Matching [10].
- *Number of measures used*. Different applications demand different measures of matching quality. Thus, it might be useful to define domain specific measures.

## 7. Acknowledgements

We thank Fausto Giunchiglia, Erhard Rahm, Phil Bernstein, and Do Hong Hai for their useful and valuable feedback.

## 8. References

1. J. Berlin, A. Motro: Autoplex: Automated Discovery of Content for Virtual Databases. CoopIS 2001,108-122
2. J. Berlin, A. Motro: Database Schema Matching Using Machine Learning with Feature Selection.CAiSE 2002
3. P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data Management for Peer-to-Peer Computing: A Vision // Technical Report # DIT-02-013. Also in proceedings of Web DB 2002
4. H. Do, S. Melnik, and E. Rahm: Comparison of Schema Matching Evaluations, Proc. GI-Workshop "Web and Databases", Erfurt, Oct. 2002
5. H.Do, E. Rahm: COMA - A System for Flexible Combination of Schema Matching Approach.VLDB 2002
6. D. Embley, D. Jackman, L. Xu: Multifaceted Exploitation of Metadata for Attribute Match Discovery in Information Integration. WIIW 2001
7. F. Giunchiglia, I. Zaihrayeu: Making peer databases interact - a vision for an architecture supporting data coordination. Proceedings of the Conference on Information Agents, Madrid, September (2002)
8. C. Ghidini, F. Giunchiglia Local Models Semantics, or Contextual Reasoning = Locality + Compatibility // Artificial Intelligence. 127(3):221-259, 2001
9. F. Giunchiglia Contextual Reasoning // IRST Technical Report # 9211-20, Trento, Italy, 1997
10. F. Giunchiglia, P. Shvaiko Semantic Matching // Technical Report # DIT-03-013
11. F. Giunchiglia, P. Shvaiko, M. Yatskevich Archive of matching examples. http://www.dit.unitn.it/~p2p/Experimentaldesign.html
12. W. Li, C. Clifton: Semantic Integration in Heterogeneous Databases Using Neural Networks. VLDB 1994
13. W.Li, C. Clifton, S.Y. Liu: Database Integration Using Neural Networks: Implementation and Experiences. Knowledge and Information Systems 2: 1, 2000
14. J. Madhavan, P. Bernstein, E. Rahm: Generic Schema Matching with Cupid. VLDB 2001
15. S. Melnik, H. Garcia-Molina, E. Rahm: Similarity Flooding: A Versatile Graph Matching Algorithm. ICDE 2002

16. R.Miller, et al. The Clio Project: Managing Heterogeneity. SIGMOD Record 30:1: 78-83, 2001
17. E. Rahm, P. Bernstein: A Survey of Approaches to Automatic Schema Matching. VLDB Journal 10: 4, 2001
18. L. Serafini, F. Giunchiglia, J. Mylopoulos, P. Bernstein. The Local Relational Model: A Logical Formalization of Database Coordination // IRST Technical Report 0301-08 To appear in Proceedings of CONTEX'03

## Appendix: Schemas used in the evaluation

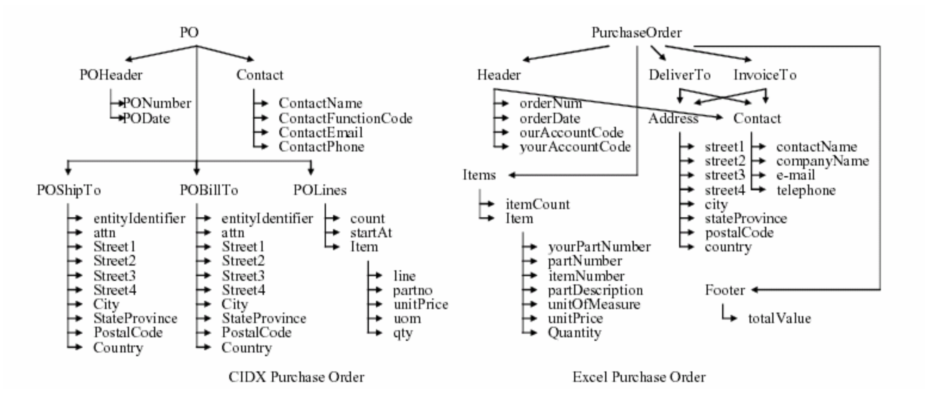


**Fig. 5.** CIDX_EXCEL. Two purchase order XML schemas.

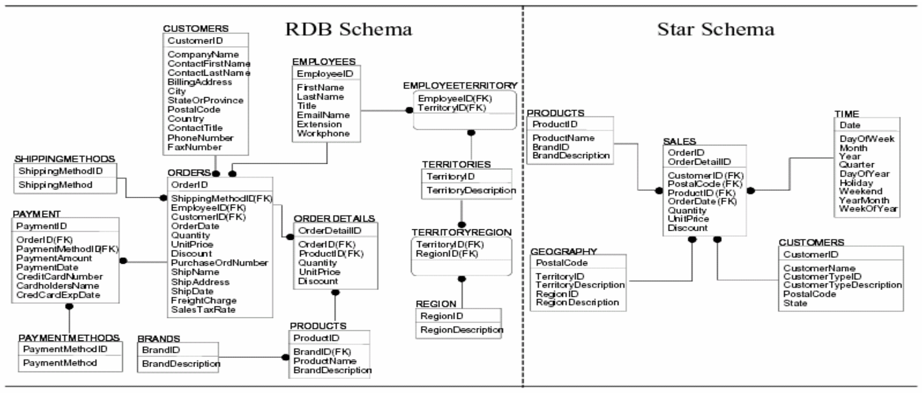
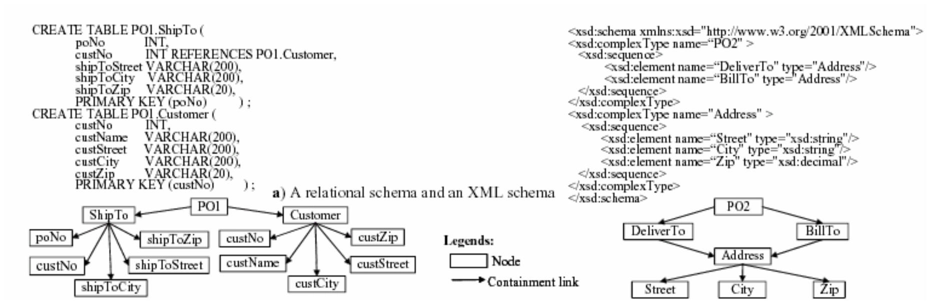


**Fig. 6.** RDB_STAR. RDB and. Star data warehouse schema



```
CREATE TABLE PO1.ShipTo (
        poNo          INT,
        custNo        INT REFERENCES PO1.Customer,
        shipToStreet VARCHAR(200),
        shipToCity    VARCHAR(200),
        shipToZip     VARCHAR(20),
        PRIMARY KEY (poNo)          ) ;
CREATE TABLE PO1.Customer (
        custNo        INT,
        custName      VARCHAR(200),
        custStreet    VARCHAR(200),
        custCity      VARCHAR(200),
        custZip       VARCHAR(20),
        PRIMARY KEY (custNo)        ) ;
```

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:complexType name="PO2" >
  <xsd:sequence>
       <xsd:element name="DeliverTo" type="Address"/>
       <xsd:element name="BillTo" type="Address"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Address" >
   <xsd:sequence>
      <xsd:element name="Street" type="xsd:string"/>
      <xsd:element name="City" type="xsd:string"/>
      <xsd:element name="Zip" type="xsd:decimal"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

**Fig. 7.** Simple. PO schemas: RDB schema vs. XML schema.

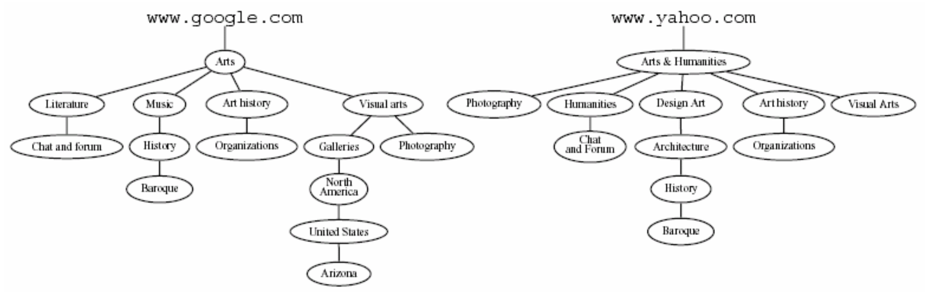


**Fig. 8.** Google Yahoo. Two concept hierarchies, namely parts of Google and Yahoo web directories.