02-AS

f=U

The 6th World Multiconference on Systemics, Cybernetics and Informatics

July 14-18, 2002
Orlando, Florida, USA

PROCEEDINGS

Volume VII

Information Systems Development II

Organized by IIIS
International
Institute of
Informatics and
Systemics

Member of
International Federation of
Systems Research IFSR

EDITED BY
Nagib Callaos
John Porter
Naphtali Rishe

On the Algorithm for Semantic Wrapping of Relational Databases*

Naphtali D. RISHE, Maxim V. CHEKMASOV, Rosany H. RODRIGUEZ,
Scott C. GRAHAM, Daniel J. MENDEZ

High Performance Database Research Center
School of Computer Science
Florida International University
Miami, Florida 33199, U.S.A.

ABSTRACT

It is well-known that the relational database model and the SQL query language are presently the most popular tools to implement and query databases. A certain level of expertise is needed to use SQL. Thus, it is a common practice to create numerous applications atop relational database aiding users to query the database and receive results. In the present paper we discuss and illustrate an algorithm to represent the relational database schema in a semantically rich way. Use of the resulting semantic schema (also called semantic view of the relational database schema) in database applications substantially reduces the size and complexity of queries to the relational database, shortens database application development cycle, and improves maintenance and reliability by reducing the size of application programs.

Key words: Semantic Wrapper, Semantic SQL, Schema Transformation, Relational Databases

1. INTRODUCTION

An algorithm describing the translation process of a relational database schema into a semantic schema, which is an essential part of the semantic wrapping methodology, is thoroughly discussed in [1]. The semantic wrapping methodology, in turn, is based on the semantic modeling approach proposed in [2]. The algorithm under consideration facilitates creation of a high-level semantic database design, which is free of the technicalities and complications of the relational database schema. This algorithm is implemented as a component of a software tool, called Semantic Wrapper. Semantic Wrapper provides access to the relational databases by a broader audience of users by employing Semantic SQL for querying semantic views rather than the relational database directly. We refer to [3] for details on Semantic SQL. Furthermore, the use of the mapping algorithm presented here is extended in the design of a heterogeneous multi-database environment, which includes relational, semantic databases and Internet data sources, as explained in [1]. Throughout the paper we use the term 'semantic schema' as it is introduced in [2]. By relational database schema we assume a set of objects (like tables, primary/foreign key constraints, etc.) created under a certain RDBMS (relational database management system) to serve users' data storage and retrieval needs.

2. DESCRIPTION OF THE ALGORITHM

The process of creating a semantic view of a relational database assumes that metadata describing tables, their respective attributes, primary/foreign keys and other constraints can be received via RDBMS hosting the database. This information is used to map relational constructs into the concepts of a semantic schema. The mapping algorithm does not affect the data, which is stored in the relational database. The algorithm for creating a semantic view of the relational database can be described as follows:

(1) For each table in the relational database create a category on the semantic view.
(2) For each column in the relational table create an attribute in the corresponding category on the semantic view.
(3) For each functional dependency in the relational database create a relation on the semantic view.
(4) Since relations are established between the categories, remove the attributes which represent foreign keys from the domain categories of the relations on the semantic view.
(5) Replace the categories that represent tables serving the role of many-to-many relationship between other tables in the relational database with many-to-many relations between the corresponding categories on the semantic view.
(6) Replace the categories that represent tables serving the role of recursive reference in the relational database with the relation of type is-part-of in the corresponding category on the semantic view. The cardinality of this relation may be many-to-one or many-to-many depending on the original relationship implemented in the database.

* This research was supported in part by NASA (under grants NAG5-9478, NAGW-4080, NAG5-5095, NAS5-97222, and NAG5-6830), NSF (CDA-9711582, IRI-9409661, HRD-9707076, and ANI-9876409), ONR (N00014-99-1-0952), and the FSGC.

(7) Replace the categories that represent tables serving the role of one-to-many relationships in the relational database with the attributes of cardinality one-to-many in the corresponding category on the semantic view.
(8) Introduce subcategory/supercategory hierarchy on semantic view.

Note that performing steps (1)-(4) of the algorithm will already lead to a valid semantic view of the relational database. Steps (5)-(8) lead to further refinement and enhancement of the semantic view. Thus in practice, steps (5)-(8) of the algorithm are not mandatory but are highly recommended. Implementing all the steps of the algorithm will usually lead to the semantic schema that shows the semantics of the data in the database in a most efficient way.

3. ILLUSTRATION OF THE ALGORITHM

We will now turn to Figure 1 to illustrate the algorithm of creating a semantic view for the relational database.

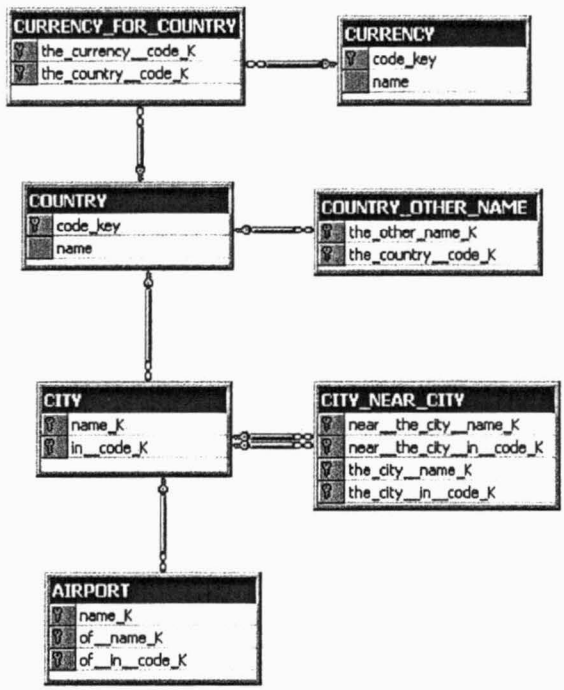


Figure 1. Relational schema of Geography database.

Figure 1 shows a Geography database which stores information about airports, cities, countries, proximity of cities to each other, and currencies used in the countries. There are several requirements to the database that led to the current implementation of the relational schema. The first requirement mandates that the country may use more than one currency. For example, The United Kingdom uses British Pounds (GBP) and Euro of the European Union (EUR). Another agreement is that the database stores not only official names of the country, but also other commonly used names. For example, the database may store 'The United States of America', 'The United States', and 'The U.S.A.' for this country. Finally, the database should relate the cities that are located relatively close to each other. This may help travelers to identify cities they may take a flight to. For example, a person traveling to Miami may arrive at Miami International (MIA) or Ft. Lauderdale International (FLL) airport. Let us now use the algorithm described above to create a semantic view for the database.

Steps (1)-(4) of the algorithm are relatively straightforward and will lead us to the semantic view presented on Figure 2.

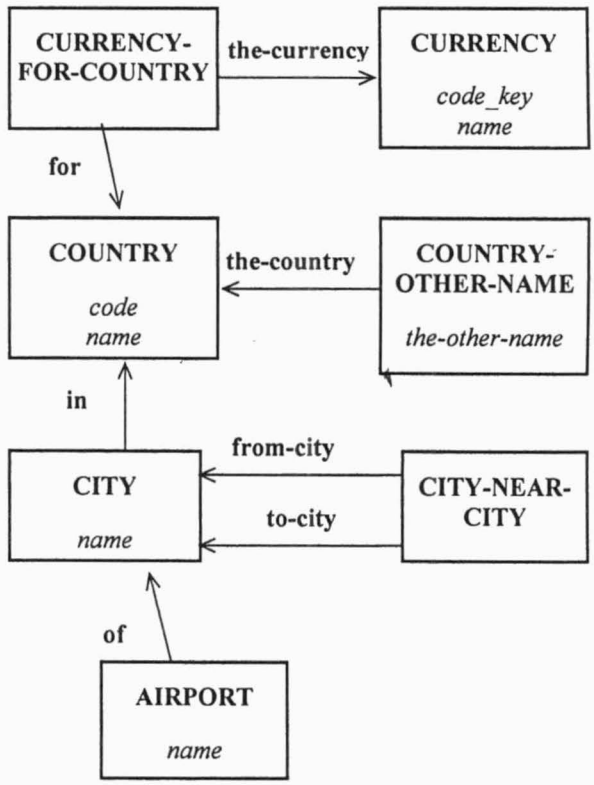


Figure 2. Intermediate semantic schema for Geography database.

Having the intermediate semantic schema of the database we can implement step (5) of the algorithm. We recognize that category **CURRENCY-FOR-COUNTRY** has no attributes and it is a domain of two many-to-one relations **the-currency** and **for** to the categories **CURRENCY** and **COUNTRY** respectively. It is evident that **CURRENCY-FOR-COUNTRY** does not have any additional information to what we have in the categories **CURRENCY** and **COUNTRY** and that it solves solely the purpose of many-to-many relationship between the respective categories. As a result of step (5) of the

algorithm we replace **CURRENCY-FOR-COUNTRY** with the many-to-many relation **for** between **CURRENCY** and **COUNTRY** as indicated on Figure 3.

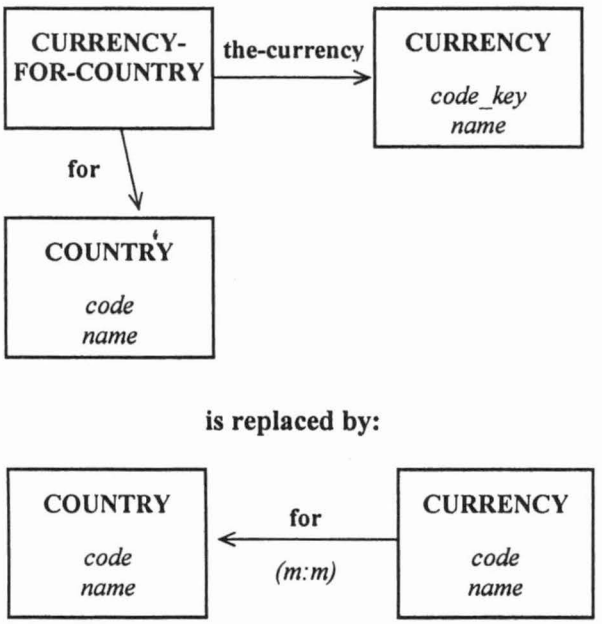


Figure 3. Application of step (5) of the algorithm to Geography database.

We have the case on our intermediate semantic view to employ step (6) of the algorithm. As we see on the schema the category **CITY-NEAR-CITY** has no attributes. However the important elements of the schema are two relations **from-city** and **to-city** of cardinality many-to-one from **CITY-NEAR-CITY** to **CITY**. They represent the information on proximity of cities to one another. At this stage we can eliminate category **CITY-NEAR-CITY** with its relations and replace it with the relation **near** of type is-part-of with cardinality many-to-many in category **CITY** as illustrated on Figure 4.

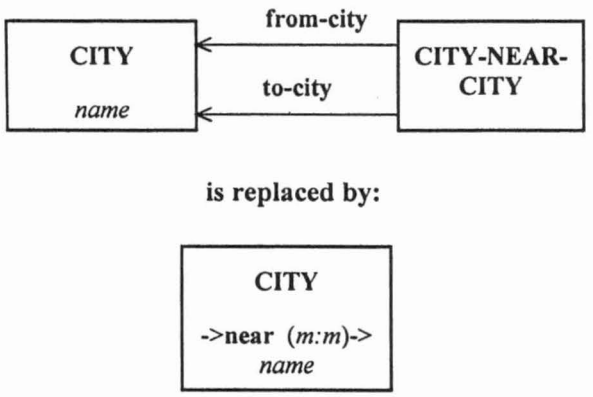


Figure 4. Application of step (6) of the algorithm to Geography database.

With respect to step (7) of the algorithm we also have a good candidate among the categories to work with. As we see on the intermediate semantic view categories **COUNTRY** and **COUNTRY-OTHER-NAME** essentially store the same information, the names of the countries. Thus there is little sense to keep two categories on the semantic view with the same information. We may introduce an additional attribute *other-name* to the category **COUNTRY** and get rid of the category **COUNTRY-OTHER-NAME** on the semantic view. However, since relation **the-country** of cardinality many-to-one has **COUNTRY-OTHER-NAME** as its domain and **COUNTRY** as it's range the cardinality of a new attribute *other-name* in **COUNTRY** will become one-to-many. Figure 5 illustrates the transformation.

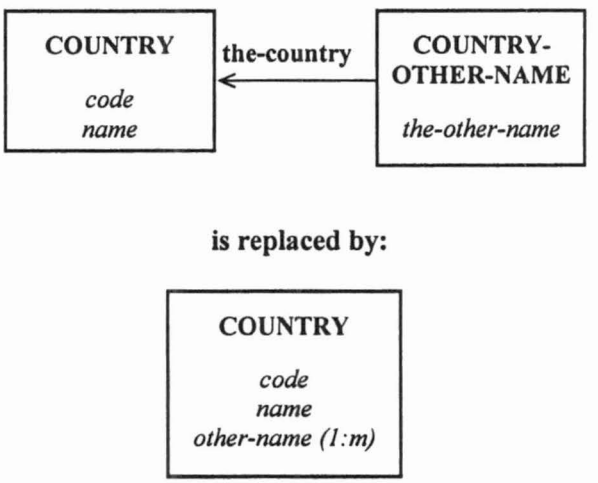
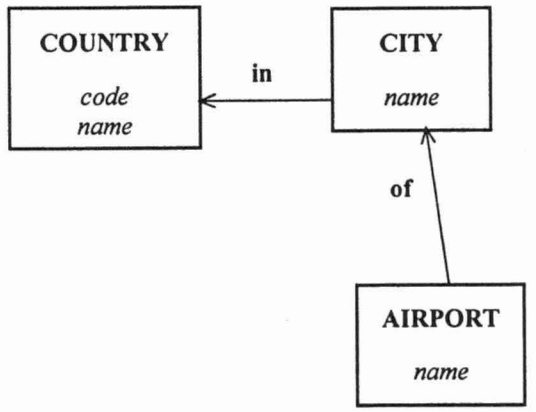


Figure 5. Application of step (7) of the algorithm to Geography database.

Finally, by employing step (8) of the algorithm we may reach one more level of abstraction on the semantic view of the relational database. We notice that countries, cities and airports are geographical entities that have a common feature: the objects of the categories **COUNTRY**, **CITY** and **AIRPORT** should have names. This allows us to introduce a supercategory for these categories, which will store this common attribute. Thus we introduce category **GEOGRAPHICAL-ENTITY** with one attribute name. The respective categories **COUNTRY**, **CITY** and **AIRPORT** will now be the subcategories of the new category and attribute name will be deleted from these three categories. Figure 6 illustrates this step.



is replaced by:

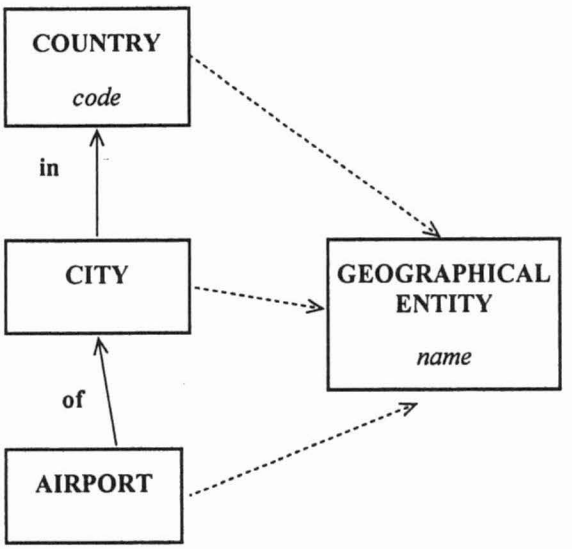


Figure 6. Application of step (8) of the algorithm to Geography database.

4. DISCUSSION ON SEMANTIC VIEWS

We have learned how to create a semantic view of the relational database, but the main goal is to use the view to query the database. In this section we would like to discuss the questions related to creation, maintenance and use of the semantic views when they are supported by a real-time software system.

We have performed research on how many steps of the semantic view creation algorithm can be automated for a conventional relational database implemented on a popular commercial RDBMS such as Oracle, Microsoft SQL Server, IBM DB2, or Sybase. It is well-known that although commercial RDBMS support the data definition language (DDL) of the SQL standard, they usually enhance the DDL for their respective databases. This allows the RDBMS developers to fully exploit the benefits of their particular implementation of the relational database model, but it makes it harder for our proposed system, Semantic Wrapper, to read the relational database metadata and to use that information to create a default semantic view. We have identified that steps (1)-(4) of the algorithm can be automated for a conventional relational database. Steps (5)-(8) will usually require human intervention.

We have identified that the DBA (database administrator) of the relational database should be the person responsible for creation and maintenance of the semantic views. There are several reasons to support this proposition:

(i) The DBA possesses a high level of expertise on the relational database model; she will easily understand the principles of the semantic view creation.

(ii) The DBA is supposed to be the most knowledgeable person on the data objects stored in a particular database and the relationships between data objects. This will help her to create the most accurate semantic view of the database.

(iii) It is the responsibility of the DBA to create and maintain userviews for different groups of database users. A semantic view of the relational database may be treated as a userview describing the semantics of the whole database. For security or other reasons the DBA may decide to create different semantic views to tailor needs of different database applications or users. If we take the Geography database as an example, a separate userview having only categories **CURRENCY** and **COUNTRY** may be created for a currency conversion application. Another userview with only categories **AIRPORT** and **CITY** will serve the application, searching nearest airports for the city, and so on.

(iv) Overall, the DBA is responsible for maintenance and correct functioning of the database tools. The proposed system implementing semantic wrapping technology, namely Semantic Wrapper, may be viewed as an additional database tool in hands of the DBA and the users.

Another problem which needs to be discussed is the extent to which the Semantic Wrapper tool should allow the DBA to modify semantic views. No questions arise when the DBA drops categories and relations from a semantic view of the database, say to create a separate view for a particular group of users. But step (8) of the algorithm allows the addition of new categories to the semantic views in the role of supercategories for the

existing categories. This may eventually lead to creation of additional relations on the semantic view as well since the subcategories may have the same relation(s), which will be propagated to the supercategory. Thus, there is a possibility that the DBA will be able to create arbitrary categories and relations that are not related to any data objects in the relational database. This in turn may lead to incorrect semantic queries against the relational database thus affecting the whole integrity of the semantic wrapping process.

This problem is addressed by enforcing the following rule: At any step of semantic view creation process, Semantic Wrapper keeps the mapping information between the relational database tables and the corresponding semantic view categories/relations intact. If modifications of the semantic view will damage the integrity of the mapping information, it is not allowed by the system.

With respect to the mapping information between the relational database and its semantic view, it is also used to translate semantic queries to the analogue relational queries each time the user or application poses a query against the database. It is natural to have a database or other easily accessible and updateable storage to store this mapping information.

5. CONCLUSION

In order to adopt and successfully use the semantic wrapping approach in the relational database environment, the DBA needs to define a semantic view for the database. This effort depends on the degree of complexity of the database and the quality of its existing documentation. If the database is well documented and has a conceptual schema in place, then the effort is relatively small. Otherwise, the DBA needs to reverse engineer the existing database schema. As a byproduct, the reverse engineering effort creates documentation on the database schema, which facilitates database use, improves its reliability, helps in training personnel, and allows the executive managers to better understand their data. With translation from the relational database schema to the semantic database schema and with Semantic Wrapper we propose an algorithm and tool that mechanizes the reverse engineering and semantic wrapping process to the degree possible, and provides a means for the DBA to make manual improvements.

6. REFERENCES

[1] R. Rodriguez, Semantic Wrapping for Heterogeneous Database Management, PhD thesis, Florida International University, 2002.

[2] N. Rishe, Database Design: The Semantic Modeling Approach, McGraw-Hill, 528 p., 1992

[3] N. Rishe, Semantic SQL, internal document, High Performance Database Research Center, Florida International University, Miami, Florida, 1998.