Querying The Software Information Base

(Extended Abstract)

Gerd Hillebrand* Polivios Klimathianakis†

1 Introduction

The Software Information Base (SIB) [1] was developed at FORTH as a repository system for re-usable software components. Originally intended to support the development of very large software systems within the ESPRIT ITHACA project, it has since been adapted to other application domains and in its current version (known as the Semantic Index System) provides a general tool for documenting and indexing large collections of interrelated heterogeneous data such as engineering designs, museum collections, organograms, etc.

The SIB consists of a knowledge-base component built upon the TELOS knowledge representation language [5] and a collection of user and application interfaces, among them a TELOS parser, a graphical browser, static analyzers for various common programming languages, an interface to relational DBMS, and others. Information is entered into the system either automatically, e.g. by loading it from a relational database or by static analysis of a program module, or manually by means of a forms-based entry facility. Once the data is in the system, the user may browse it using the graphical interface or run a set of certain pre-defined queries against it.

In this paper, we present the design of an ad-hoc query facility for the SIB, which allows the user to write queries in a high-level, SQL-style language and have the system execute them. Our goal was to build a query language that was close to existing relational DB languages, sufficiently powerful to express fairly sophisticated queries, and easily optimizable. A first version of the query language has now been specified, and implementation of the query processor is under way.

2 Telos

The data model of the SIB is that of TELOS [5]. (In fact, a simplified version of TELOS without temporal assertions is used.) A complete axiomatization of the TELOS semantics is given in [3]. Informally, there are two kinds of objects in TELOS: *individuals*, which model real-world entities, and *attributes*, which model binary relationships between them. Both are treated uniformly by the language. Objects are organized along three dimensions: *attribution* (linking objects via attributes), *classification* ("instance-of" links) and *generalization* ("isa" links). TELOS provides for an infinite instantiation hierarchy and a weak typing of attributes by means of attribute classes. Individuals and attributes are referred to by logical names, where the name of an individual must be unique across the entire database and the name of an attribute must be unique among all attributes of its source object (a more sophisticated scoping mechanism is under development).

A TELOS database may be viewed as a directed graph, where the nodes are labeled with names of individuals and the edges are labeled with either attribute names or "instance-of" or "isa". Attributes may themselves have attributes, so edges may originate from other edges, but this case is not very common in practice. However, there are usually "isa" and "instance-of" links between attribute objects. The task of a high-level query language is to allow easy and efficient navigation of this graph.

*Contact author. Informatics Department, Rutherford Appleton Labs, Chilton, DIDCOT, Great Britain OX11 0QX. Phone: +44 (235) 44-5710. Fax: +44 (235) 44-5831. Email: ggh@inf.rl.ac.uk. This work was supported by an HCM postdoctoral fellowship and was done while the author was visiting FORTH.

†Institute of Computer Science, Foundation of Research and Technology - Hellas, P.O. Box 1385, Heraklion, Crete, Greece 71110. Email: polivios@ics.forth.gr.

3 Language Design and Implementation

After an analysis of the queries commonly used in various SIB applications, we identified the following requirements for a high-level query language:

- the language should be declarative in style, intuitive to use, and preferably close to some well-known existing language, because of a large non-technical user base,
- navigation within the semantic net must be easy,
- queries using meta- and higher order classes and queries to the schema must be supported,
- transitive closure queries should be supported,
- queries should be parametrizable for re-use,
- the language must be amenable to optimization.

We decided to meet these requirements by choosing an SQL-style language augmented with path expressions a la XSQL [4], transitive closure operators, and a compile-time name-to-sysid translation a la ConceptBase [2]. Queries are built from select-from-where expressions, where the predicates in the "where" clause are path expressions of the form $X.attr_1.attr_2.\ldots.attr_k[Y]$. In their most general form, path expressions are regular expressions over attribute categories, meta-categories, etc., and variables. The semantics of the language is nevertheless strictly first order. Literals occuring in path expressions are resolved into system identifiers at compile time using class information provided in the "from" clause, so that compiled queries refer only to concrete objects and, for attribute objects, their source and destination objects. This kind of information can be very efficiently retrieved from the SIB because of the way objects are stored internally.

Queries are processed in the system as follows. A parser analyzes the SQL syntax, translates literals into object identifiers, and translates the query into a set of Datalog rules. The predicates in these rules correspond to attribute class objects, which may be annotated with transitive closure operators (such closures are handled by specialized operators already available in the SIB application interface). An optimizer then rewrites these rules according to a simplified magic-sets transformation. Finally, a code generator translates the rewritten rules into a set of calls to the SIB application interface. At this point, the generated code may be stored for later re-use (with possible substitutions for the system identifiers appearing in the code), or immediately executed.

References

[1] P. Constantopoulos, M. Jarke, J. Mylopoulos, Y. Vassiliou. *The Software Information Base: A Server for Reuse.* FORTH-ICS technical report, 1993.

[2] M. Jarke, S. Eherer, R. Gallersdörfer, M. Jeusfeld, M. Staudt. *ConceptBase - A Deductive Object Base Manager.* RWTH Aachen technical report TR 93-14, 1993.

[3] M. Jeusfeld. *Change Control in Deductive Object Bases.* Dissertation, Universität Passau, 1992.

[4] M. Kifer, W. Kim, Y. Sagiv. Querying Object-Oriented Databases. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pp. 393–402, 1992.

[5] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis. Telos: Representing Knowledge About Information Systems. *ACM Transactions on Information Systems* **8** (1990), pp. 325–362.