An Ontology Tool for Query Formulation in an Agent-Based Context

Kuhanandha Mahalingam and Michael N. Huhns
Center for Information Technology
Department of Electrical and Computer Engineering
University of South Carolina
Columbia, SC, U.S.A. 29208
(803) 777-5921 or kuha@sc.edu
(803) 777-8045 FAX

Abstract

This paper describes how query formulation can be made simple and less complicated by using ontologies. It takes a brief look at several advantages of using ontologies in a distributed, heterogeneous, and dynamic information environment, such as the Internet. It also examines the construction and evaluation of an ontology-based distributed information system developed using the Java language. It discusses issues related to software tools that operate in a distributed environment, and shows how the client-server architecture and the agent technology used by the Java language can be used effectively in such environments. The software is applied to an information system for healthcare administrators, which spans hospitals, clinics, and governmental health departments.

1. Introduction

An approach to achieving interoperation among distributed and heterogeneous information sources is to introduce software agents to serve as mediators, translators, and information brokers – this is the essence of a cooperative information systems approach. The major task for the agents is to reconcile the varied semantics of the mostly autonomous resources. We have developed a tool that not only allows the user to construct and browse ontologies, but also enables query formulation at a very abstract level where novice users are more comfortable. This paper describes this tool and its use in a representative application, healthcare information systems.

1.1 Use of Ontologies as a basis for achieving interoperation

As a consequence of the Internet's rapid growth, its users are faced with many new problems. One of the most frequently encountered problems is how to search for and retrieve necessary information from the large number of information sources available on the Internet. The data provided by the sources is no longer just simple text; with the introduction of multimedia and new technologies, it has become much more complex than before. As a result, old methods for manipulating these sources are no longer appropriate or even efficient. In order to evolve with the growth of the Internet, we need not only suitable storing and retrieval mechanisms, but also efficient search tools that can harvest the necessary information from these sources. These mechanisms should allow efficient querying on diverse information sources that support structured as well as unstructured data. In such complex and heterogeneous environments, ontologies are best suited for information storage and retrieval. In contrast to unstructured data, data stored as ontologies can capture the structure in addition to the semantics of information spaces. We believe that in a distributed and heterogeneous environment such as the Internet, ontology-based manipulation of these diverse sources is the most desirable solution for achieving interoperation

1.2 What is an ontology

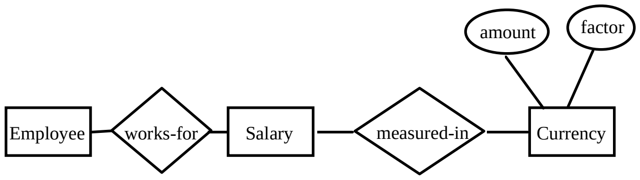
From an artificial intelligence viewpoint, an ontology is a model of some portion of the world and is described by defining a set of representational terms. In an ontology, definitions associate the names of entities in a universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms [2]. In essence, ontologies can be used very effectively to organize keywords as well as database concepts by capturing the semantic relationships among keywords or among tables and fields in a relational database. By using these relationships, a network structure can be created providing users with an abstract view of an information space for their domain of interest. Ontologies are well suited for knowledge sharing in a distributed environment where, if necessary, various ontologies can be integrated to form a global ontology.

1.3 Advantages of using ontologies

1.31 Provisions for value mapping

Using ontologies has an added advantage over unstructured text-based information spaces in terms of value mapping. Currently there are no tools that can map values from one unit to another, so that when a result set is returned the user knows the corresponding units for that set. For example, when the salaries of employees are requested and the results are returned, the result set does not provide any information about whether the salaries are in dollars or pounds or both. It is usually assumed to be the default currency used in the country of concern. However, accepting the default is not always safe, particularly given the distributed nature of the environment, where a query might be generated in one country with one unit of currency and the result might come from another country with a different unit of currency. In a distributed and heterogeneous information environment, there are many such problems in mapping values of data from one to another.

With an ontology, there are several ways to find a suitable solution to this problem. One such solution is described in Figure 1 below, where “factor” is used as an attribute of the entity “salary” that is used to map one currency to the other. Thus, when the result is returned, the value of the attribute “factor” would indicate the currency unit used in the result set and the system or person who made the query can easily convert the results to the desired currency. This advantage in using ontologies is a very desirable feature in a distributed environment.



currency name	factor
US $	1.00
DM	0.62
others..	…

Figure 1. Value mapping example using Employee-Salary relationship

1.32 Suitable for graphical representations

Textual representation of the ontologies vary depending on the environment where they will be used. The two most popular representations are the Entity-Relation (ER) and the Frame-slot (FS) models. In the ER model a concept is represented by an "entity" and the attributes of that concept are represented by "attributes." Entities may be related by one or many "relations." In the FS model, both concepts and relationships among those concepts are represented by "frames" and the attributes of a given concept is represented by "slots" of that particular "frame." In spite of their simplicity, these textual representations are not so easy for an average user to comprehend. But on the contrary, a graphical representation of the same ontology can be very easily understood by any one regardless of their expertise in the field. Ontologies can very clearly be represented in a graphical format by using the ER model.

1.33 Suitable for query formulation

One of the most frequently encountered problems in a cooperative environment is getting the necessary information from the vast amount of information sources available on the Internet. In other words formulating efficient queries so that the user can get the needed information. This task is easy if the user have experience in dealing with databases and know at least one of many database query languages like SQL but in reality as we all know that it is only an ideal situation. The ideal solution to this problem is to make query formulation as in a regular English statement like "get me the phone-numbers of all the Johnsons in the Columbia area." But as we mentioned, it is only an ideal solution. Our goal is to find somewhat of a middle ground that is not too specific as in SQL and not too general as in the English statement. In other words the above query could be rewritten somewhat differently as follows:

```
Person
        phone-number = <?.request.?>
        last-name = 'Johnson'
        lives-in
                City
                        name = 'Columbia'
```

As you can see, even though this format is quite identical to the regular statement, it can easily be translated to a SQL statement. The goal here is to allow the user to generate queries without the need to learn a new language such as SQL since we can not expect all users to be computer programmers. By using our tool, any user can easily generate above query by just a few mouse clicks on the displayed ontology and then submit it to the database to get required information. Our tool also allow those users who are knowledgeable in SQL to edit or fine tune these queries if and when they desire. In addition the users also have the capability to save and use these queries at a later time in various different ways by simply editing existing queries. This type of query formulation wouldn't be possible if not for graphical representation of the ontologies.

1.33 Ability to view at various abstraction levels and ability to scale

Ontologies can grow and shrink as necessary based on the context where they are being used. In a different context, part of one ontology can be hidden or another made visible so that a new view of the same information space can be generated to suit a certain group of audience. In large databases, this process need to be executed many times. Using ontologies it can be done efficiently and faster. In addition, sub-ontologies created by experts from a variety of fields, can be merged with little effort to create one super-ontology.

2. Background

There have been several attempts to accomplish the task of implementing distributed ontology-based information systems. These include MCC's Carnot, Cyc, and InfoSleuth projects, Stanford University's Ontolingua, and SRI International's GKB (Generic Knowledge Base) Editor.

The Carnot [1] project was initiated in 1990 with the goal of addressing the problem of logically unifying physically distributed, enterprise-wide, heterogeneous information. The Model Integration Software Tool (MIST), developed as part of this project, is a graphical user interface that assists a user in the integration of different databases via a common ontology that serves as an enterprise model.

The Cyc project is an ongoing attempt at building a large-scale knowledge base. It was begun as ten-year research initiative by Doug Lenat in 1984, at MCC. The Cyc common-sense knowledge base is the result of a large effort to encode a general ontology of the world, along with the rules that govern the common-sense relationships among the components of the ontology [5]. The primary task of the project is codifying a vast amount of knowledge that is considered as "consensus reality" – the background knowledge possessed by a typical person. In addition to the encoded "common-sense" knowledge, the Cyc system contains a wide range of reasoning mechanisms for the purpose of generalized deduction and analogical inference.

Ontolingua [2] is a set of tools, written in Common Lisp, for analyzing and translating ontologies developed by the Knowledge Systems Lab (KSL) at Stanford University. It uses KIF [6] (Knowledge Interchange Format) as the interlingua and is portable over several representation systems. It includes a KIF parser and syntax checker, a cross reference utility, a set of translators from KIF into implemented representation systems, and a HTML report generator.

The GKB-Editor [3] (Generic Knowledge Base Editor) developed by SRI International, is a tool for graphically browsing and editing knowledge bases across multiple frame representation systems in a uniform manner. It offers an intuitive user interface, in which objects and data items are represented as nodes in a graph, with the relationships between them forming the edges. Users edit a KB through direct pictorial manipulation, using a mouse or pen. An incremental browsing facility allows the user to selectively display only that region of a KB that is currently of interest, even as that region changes.

The InfoSleuth project [7], based on MCC's Carnot technology, was created with the intention of developing and deploying technologies for finding information in corporate and in external networks. The InfoSleuth architecture is a collection of agents that represent users, information sources, ontologies, and query engines, and that cooperate in finding and fusing information.

3. Java Ontology Editor (JOE)

JOE is a software tool, written in Sun's Java language and developed using the Microsoft's Visual J++ developer studio, that (a) provides a graphical user interface (GUI) to represent ontologies and (b) provides another GUI where queries can be created by the point-and-click approach. Unlike other languages, Java has many advantages when used in a distributed environment of autonomous and heterogeneous information resources, which characterizes our application domain—the healthcare industry.

3.1 Group editing

Because Java applets can be downloaded any where at any time as long as the user has access to a Java compatible browser, the same ontology can be simultaneously viewed and edited by more than one user. This group editing feature has many advantages. It saves storage space since several users can work on a single copy. At the same time, each user can make additional copies for their individual use if and when they desire. It also eliminates the problem of keeping different copies of the same ontology up to date. Another advantage is that several people can jointly build an ontology over a length of time. In addition, if the need arises, users can easily merge various sub-ontologies to create one large super-ontology. This feature allows people with different expertise to cooperate in creating one global ontology as in large enterprises where such joint ventures take place on a regular basis.

3.2 JDBC (Java Data Base Connectivity (JDBC)

Sun's JDBC support provides an added advantage over other languages in a distributed and heterogeneous environment. Sun's JDBC-ODBC bridge allows Java applets to communicate with most commercially available database drivers such as MS Access, Sybase, and Oracle through the Microsoft's ODBC (Open Data Base Connectivity) driver. This means that users only need to support one standard query language and be less concerned with others. In addition, several compatibility issues resulting from using different computer platforms and different software vendors can also be resolved without too much overhead.

4. System Architecture of JOE

4.1 Applet framework

JOE is a graphical user interface that has the following two major parts: (1) an ontology editor and (2) a query formulation tool as shown below in Figure 2. The Ontology Editor provides a user interface where a user can create a new or edit an old ontology by adding new concepts (entities), attributes for that concept and relationship between two or more concepts. The query formulation tool is also a user interface that allows a user to build queries on the information space that is displayed on the Ontology Editor.

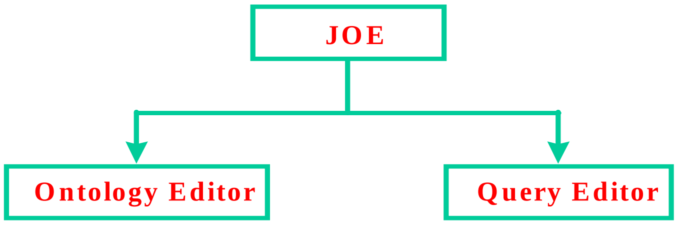


Figure 2. The applet architecture for JOE

4.2 Abstraction Mechanisms

Graphical ontology editors typically do not work well with a large number of nodes or with a large number of links (arcs) among them as a result of the limited viewing area on most computer monitors. In addition, navigating among a large number of nodes and links is not an easy task. In order to find some feasible solutions to these kinds of problems we added few new features in JOE targeted mainly toward useful abstraction mechanisms:

- *Selective viewing* - JOE allows the user to view the Ontology with complete details or if desired, with only selected types of nodes in the ontology. In other words, the user can view only entities, entities and attributes, or entities and relations. This option to selectively view nodes can reduce the complexity and confusion involved in a large ontology. In Figure 3 below, the left side shows only the entities and attributes and the right side shows only the entities and relations.

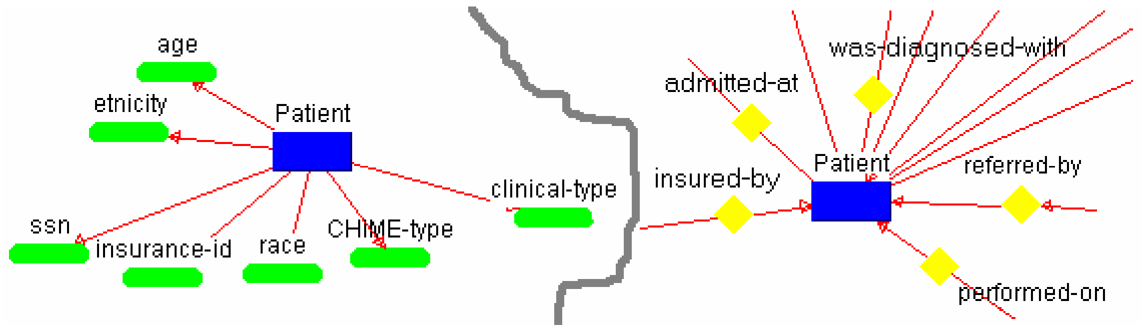


Figure 3. JOE with two snapshots of selective viewing *[entities & attributes only on the left and entities & relations on the right]*

- *Searching* - JOE provides a window, as shown on the left side of Figure 6, with a listing of all available nodes in the ontology. The user can locate a node by just double clicking on the name of that node on the displayed list, and JOE will automatically scroll the viewing window to that particular node. This option will eliminate the need for searching a specific node in a large ontology with a large number of nodes.

- *Zooming* - JOE can display entire ontology inside the current window by zooming out appropriately. This feature allows the user to view the complete ontology in full at any time. In Figure 4 below, the entire healthcare ontology is shown zoomed out to fit the current window.

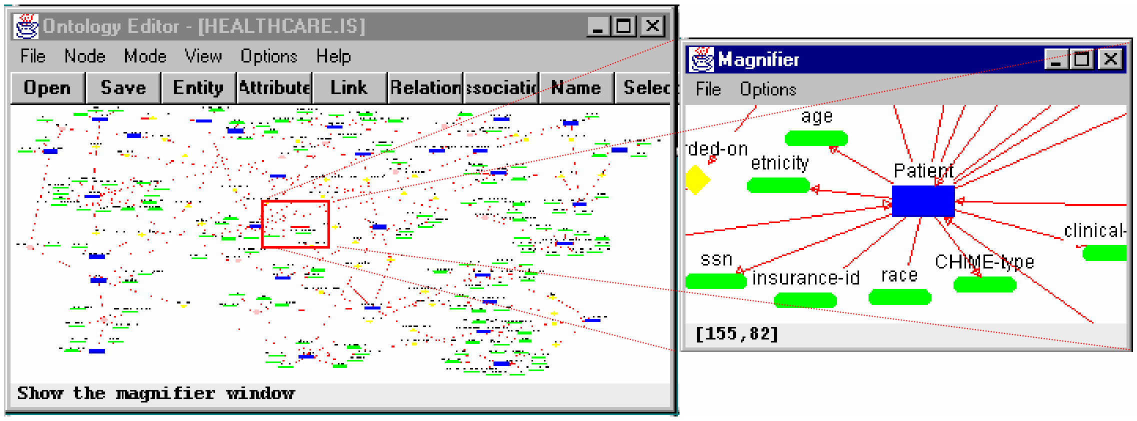

Ontology Editor - [HEALTHCARE.IS]
File Node Mode View Options Help
Open Save Entity Attribute Link Relation ssociatio Name Selec
Show the magnifier window
Magnifier
File Options
age
rded-on
etnicity
Patient
clinical-
ssn
insurance-id
race
CHIME-type
[155,82]

Figure 4. JOE displaying entire ontology within the current window and the magnified view of the selected area on the right

- *Magnification* - When the entire ontology is displayed as described above, JOE also provides a “magnifying glass” that will magnify a small portion of the ontology under the mouse. This is necessary if the ontology is quite large and detailed information can not be displayed in the zoomed-out image of the ontology. This magnified portion is displayed in a separate window, as shown in the right side of Figure 4 above, giving the user a sense of which region of the ontology he or she is viewing at that moment. If the user clicked the mouse any where inside the window, then the viewing mode will be set to normal and the window will be scrolled automatically to display that region of the ontology.

- *Hierarchical information* - JOE, provides the user with the hierarchical information of a given ontology in a tree like structure (similar to the MS Windows file manager format). This feature was added so that the user can obtain an abstract view of the given ontology if he or she desires and selectively view or work only at a level of abstraction at which they feel comfortable.

  For example, in Figure 5, a sample ontology describing different vehicles along with the class hierarchy is shown. JOE has the option to display only the expanded nodes and not the others. In this example, the user may be interested in obtaining information only about two classes, "Cars" and "Trucks" as shown below. Therefore it is not necessary to display any information about other classes in the ontology. It is not only redundant as far as the user is concerned, but also would introduce unnecessary confusion. This abstraction mechanism would help us display large ontologies in a simple easy to read format by reducing the complexity involved.

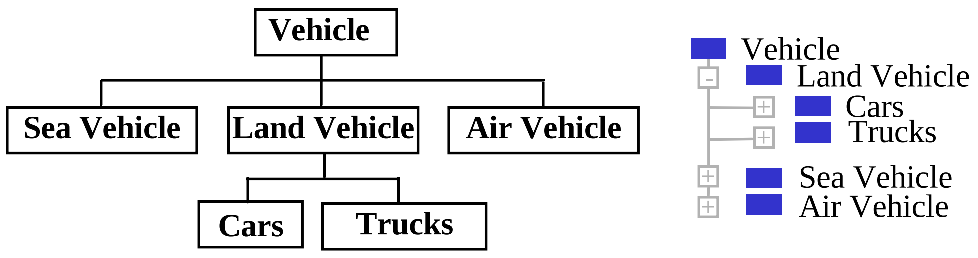


Figure 5. A vehicle ontology and its corresponding hierarchy tree

In addition to above mentioned abstraction mechanisms, JOE also supports most of the basic editing functionality such as selecting, cutting, and moving. However, the current version of JOE does not support undoing an operation, copying or pasting. These operations will be added in the near future.

5. A test application

Currently, the development of JOE is targeted towards a healthcare application: the Healthcare Information Infrastructure Technology (HIIT) project [11] funded under the Department of Commerce Advanced Technology Program.

The healthcare industry provides many opportunities in which such tools as JOE can be used. First and foremost, even though there are so many hospitals and healthcare providers all over the country, the information fundamental to all these facilities is the same no matter where they are located. Yet, it is not possible for one facility to request information from another in a straightforward manner. It is mainly because there are no globally accepted standard architectures. The architecture used to represent such information is different from one to the other and it is difficult to translate from one facility to another.

The idea behind our application is to represent, simply, the abstract view of the information fundamental to all healthcare industries by a global ontology, so that queries made on this ontology will support a uniform standard irrespective of individual healthcare facilities. These queries, eventually, would be further refined through intermediate “translating-agents” just as the Resource Agents of the InfoSleuth system mentioned above, before being processed by individual healthcare providers. By accepting a global standard, all healthcare providers will be able to communicate freely with each other, while at the same time continuing to maintain their individual information source architectures. This is not only a feasible solution but also an economical one, since the cost involved in reengineering each facility to adhere to a new standard would be very expensive.

Figures 6 and 7 show JOE executing in its editor mode and its query mode, respectively. A part of a healthcare facility information space is displayed as a graph showing the Patient table, all of its columns, and a few of its relations. On the left side of the main window a list of all tables, attributes, and relations is provided so that the user can easily go to any node in the healthcare information space just by double clicks.

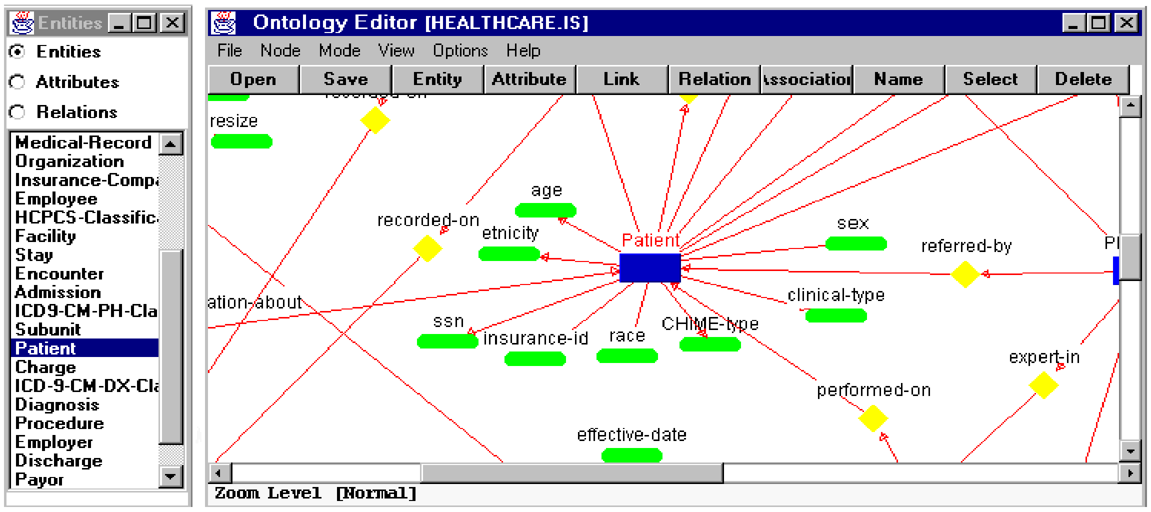
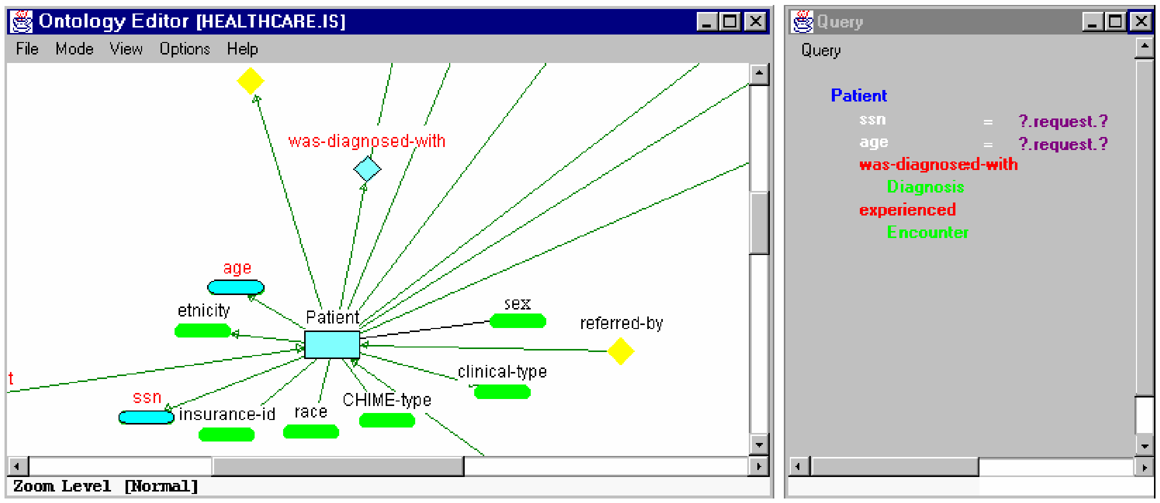


Figure 6. JOE executing in the editor mode

When executing in the query mode, JOE initially provides an option to select a “root-entity (table)” from all available entities in the currently displayed ontology. Once the root-entity has been selected, the user can build a query by any of the following combinations:

- Double clicking on the attributes (shown in green) - a dialog box will be displayed where the user can set attribute constraints.
- A single click on the relation (shown in yellow) or association (a relation with one or more attributes) shown in pink - all the entities related by this relation will be displayed for further expansion.

The current query will be displayed on a separate window to the right of the main window. All entities, attributes and relations used in the query are displayed in a different (cyan) color to indicate that they are part of the current query. The user can simply run the query by choosing the “submit” option in the “Query” menu and the results will be displayed on another separate window. JOE also provides an editor where the user can directly modify or type in a new SQL statement for execution.



```
Patient
    ssn                    =   ?.request.?
    age                    =   ?.request.?
    was-diagnosed-with
        Diagnosis
    experienced
        Encounter
```

Figure 7. JOE executing in the query mode with a partial query

As you can see, the query shown in the display can be easily translated to a regular sentence. For example, consider the following two queries:

Patient
    age = ***?.request.?***

*Here “?.request.? means that the user wants the corresponding attribute (i.e. “**age**”) from the **Patient** table in the result set.*

Patient
    age = ***?.request.?***
    has
    Medical-Condition
        condition-name = ***?.request.?***

⟷

<u>Query Statement</u>
*“get the **age** and the **name** of the **Medical-Condition** for each **Patient**”*

6. Conclusions

We believe that ontology-based representation of information spaces can be effectively used to achieve interoperation among distributed and heterogeneous information sources. Ontologies are the best tools for semantic reconciliation among software agents. Ontologies can be used to capture the structure in information sources and represent them in an easy to read format. Our tool, JOE, is an attempt to provide a adaptable software agent written in Java to create and edit such ontologies. In addition, JOE also provides a unique and easy to use query formulation tool that can function as an independent agent and has the ability to communicate with other agents via Java’s RMI protocol.

The framework of JOE provides a powerful but simple GUI that can be used to create or edit any ontology. These ontologies can be accessed by any user (even other software agents) on the Internet who may copy or modify them, depending on their individual needs. By use of Java's security features, access to such ontologies can be monitored to avoid misuse. Those users who have access to these global ontologies can make queries on them by use of point-and-click approaches. It is our intention that JOE be intuitive enough so that all users, regardless of their expertise in a given area, should be able to get desired information in an efficient manner.

References

[1] M. N. Huhns, N. Jacobs, T. Ksiezyk, W. M. Shen, M. Singh, and P. Cannata, 1992. "Enterprise Information Modeling and Model Integration in Carnot," *Enterprise Integration Modeling: Proceedings of the First International Conference*, The MIT Press. [http://www.mcc.com/projects/carnot]

[2] T. Gruber. “A translation approach to portable ontologies,” *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993. [http://www-ksl.stanford.edu/knowledge-sharing/ontolingua/ontolingua.html]

[3] P. D. Karp, K. Myers, and T. Gruber, "The generic frame protocol," in *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*, pp. 768--774, 1995. [http://www.ai.sri.com/~gkb/]

[4] K. Mahalingam, “The Java Ontology Editor (JOE),” Center for Information Technology, Electrical and Computer Engineering Department, University of South Carolina, 1996. [http://www.ece.sc.edu/Labs/hiit/html/imts-new.html]

[5] D. Lenat and R. V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project,* Addison-Wesley Publishing Company, Inc., Reading, MA, 1990.

[6] M. Genesereth. “Knowledge Interchange Format,” *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, Cambridge, MA, pages 599-600, Morgan Kaufmann, 1991. [http://www-ksl.stanford.edu/knowledge-sharing/kif/]

[7] D. Woelk, M. Huhns, and C. Tomlinson. “Uncovering the Next Generation of Active Objects,” *Object Magazine*, vol. 5, no. 4, pp. 32--40, July/August 1995. [http://www.mcc.com/projects/infosleuth/]

[8] “Modeling a Vocabulary in an Object-Oriented Database - OOHVR Query Interface Project,” New Jersey Institute of Technology Technical Report. [http://object.njit.edu:2000/~oohvr/QIP.html]

[9] Java(TM) Remote Method Invocation Specification. http://chatsubo.javasoft.com/current/doc/rmi-spec/rmiTOC.doc.html

[10] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaratnam, Y. Sagiv, Ullman, and J. Widom, “The TSIMMIS Approach to Mediation: Data Models and Languages,” *Proceedings of the NGITS (Next Generation Information Technologies and Systems),* June 1995.

[11] The Healthcare Information Infrastructure Technology Program *http://host.scra.org/hiit.html*