PARKA-DB: Back-End Technology for High Performance Knowledge Representation Systems

Kilian Stoffel
University of Neuchâtel
Neuchâtel, Switzerland

James Hendler
University of Maryland
College Park, MD USA

Abstract
In this paper we discuss a system which uses an ontology to provide access to, and loading from, a medical database system. The "pure" ontological information is used to define which individuals in the database are of interest for further analysis. The system then loads the information about these individuals into a "hybrid" ontology, one which ties the specific instances to the more general ontology. To support the large ontologies needed in this application (and a number of others), we use Parka-DB, a scalable ontology management system that uses relational database techniques to provide many desirable operational features.

1 Introduction

In the past few years ontologies have become an increasingly important aspect of research in AI. Systems such as CYC [Lenat 91] and ISI's Sensus [Knight 94] have shown the need for ever larger ontologies, and recent research areas such as Knowledge Discovery in Databases (KDD) and intelligent internet-search engines, to name but two, have shown an increasing need for the integration of semantic models with large scale data. With this growing interest, new efforts to make ontologies accessible to a larger user community, and to scale the size of these ontologies, have been undertaken. In addition, projects such as the Knowledge Sharing Effort and the Open Knowledge Base Connectivity Project [Chaudri 98], which focus on the combination of different small ontologies into larger and more complex ones, demonstrate the need for scalable ontology support tools. Unfortunately, however, most ontology-management systems do not support the extremely large ontologies needed for such projects. In this paper, we describe the Parka-DB system, a scaleable back-end technology designed to support exactly the sorts of very large ontologies demanded by these projects. First we review some of the past work in ontology development, then we briefly describe the Parka-DB system. We then describe an application in medical informatics that shows the advantage of using ontological information when interacting with the large medical databases needed for epidemiological research.

Parka-DB™ is a registered trademark of the University of Maryland.

1.1 Ontology

There is some dispute in the KR community as to what exactly an "ontology" is. In particular, there is a question as to whether "exemplars", the individual items filling an ontological definition count as part of the ontology (in this project the exemplars are the patient records). Thus, for example, does a knowledge base containing information about thousands of cities and the countries they are found in contain one assertion, that countries contain cities, or does it contain thousands of assertions when one includes all of the individual city to country mappings. While our system can support both kinds quite well, it is sometimes important to differentiate between the two. We will use the terms *traditional or pure ontology* for the former, that is those ontologies consisting of only the definitions. We use the term *hybrid ontologies* for the latter, those combining both ontological relations and the many instances defined thereon. This second group may consist of a relatively small ontological part and a much larger example base, or as a dense combination of relations and instances [Stoffel 97b].[1]

Currently, there is a significant amount of research being done in the area of ontology development and management. Most of this work can be classified in three, often overlapping, categories: efforts to create large ontologies, to define expressive languages for representing ontological knowledge, and to implement systems which support ontology-based applications.

It's not our goal to give an exhaustive list of all projects currently dealing with the creation of ontologies. Some significant examples include: efforts to create large ontology-based thesauri and dictionaries such as the *Sensus* project at ISI [Knight 94], or the *WordNet* project at Princeton [Miller 96], efforts to develop domain specific ontologies such as those used in medicine (e.g. *UMLS* [UMLS 94], *SnoMed* [Code 93]]), and efforts to populate large common-sense ontologies such as the US *CYC* project[2] [Lenat 91] and the Japanese *Knowledge Archive* project.

As well as these efforts to create specific ontologies, an important concern in ontology research is to define expressive languages, which can be used to define the ontologies. This will be very important if ontologies are to become easily accessible, reusable, and combinable. Examples of such efforts include work in *Ontolingua* [Gruber 92], *Kif* [Gruber 90], and *Conceptual Graphs* [Sowa 84] and the many modern descendants of these systems.

A third set of projects are those focusing on implementations of ontology management systems. One group of such systems are knowledge representation systems which also provide ontological support, such as *Loom* [MacGregor 94], *Classic* [Borgida 94], *CYC* [Lenat 91], *Sneps* [Shapiro 92], and *Kris* [Baader 94] (among many others). While all of these systems support their own languages, and all are very expressive, they are currently not well suited to host very large ontologies because they lack secondary memory support, database integration and other such techniques critical for scaling KR systems to extremely large ontologies (and especially to the often even larger hybrid ontologies necessary for many current applications).

---

[1] In the KR community, ontological information is sometimes divided into the T-box, which specifies taxonomical information, and the A-box, which contains instances of the specified classes. Ours is a similar distinction, but we are being more informal than the usage in the traditional KR literature.

[2] The CYC ontology is part of a large system, which includes inference algorithms and language definitions, thus spanning all three of the above categories.

Alternatively, there are some projects designed to directly examine issues in scaling KR systems such as *FramerD* [Haase 96], Lee and Geller's [Lee 96] work, *GKB* [Karp 95a], *SHRUTI* [Shastri 93] and *PARKA* [Evett 94]. All of these systems are scalable to a certain extent but most of them are still quite limited – *FramerD* is much closer to an Object Oriented Data Base System than to a KR system, *GKB* primarily relies on knowledge bases expressed in Loom or other KR systems and is thus limited by their scalability, Lee and Geller's system is limited to a restricted set of queries on specific parallel computers (the CM2 and CM5), and *SHRUTI* is very limited in the number of conjunctions it can handle in a single query. Our own earlier system, Parka, was able to scale to extremely large sizes, but required the very large memory spaces provided on parallel supercomputers.

In this paper, we focus on a medical system which uses our most recent version of Parka, called Parka-DB, which makes heavy use of database technology, relieving the need for expensive supercomputers. The new system runs on a wide variety of computer systems from laptops to high-end workstations (and it can still scale up to high-end parallel computer systems). In our work, a well-defined low-level input language enables one to write simple translators to reuse KBs/Ontologies defined for other systems. The use of secondary storage, realized by using a relational database and efficient memory management allows us to host the largest existing ontologies. The medical system allows doctors to load information about their patients' conditions (specifically microbiological data) into a hybrid ontology –using pure ontological information to load a knowledge base containing a very large number of assertions about the individuals concerned. In the remainder of the paper, we first introduce the Parka-DB system, and then describe the medical application.

2 The PARKA-DB System

2.1 Overview

Parka-DB supports a frame-based AI language (sometimes called a "property/class" system in today's literature) using high-performance and scaleable computing techniques. The goal of the project is, and has always been, to develop a fairly traditional AI language/tool that can scale to the extremely large size applications mandated by the needs of today's information technology revolution. In recent years, we've focused on the use of database technology both to support the scaling and to integrate into the more traditional data environments needed by many of the applications we are working on.

More specifically, PARKA allows the user to define a frame-based knowledge base with class, subclass, and property links used to encode the ontology. Property values can themselves be frames, or alternatively can be string, numeric values, or specialized data structures. The language allows exceptions, in the form of multiple-inheritance, and provides extremely efficient (and efficiently parallelizable) algorithms for performing inheritance using a true inferential-distance-ordering calculation (IDO, [Horty 90]).[3] Parka has also been shown to effectively compute recognition, and also to handle extremely complex "structure matching" queries -- a class of conjunctive queries relating a set of variables and constraints and unifying these against the larger KB. While it is difficult to exactly compare KR languages, a very loose categorization would put PARKA as more expressive than Classic [Borgida 94], due to

[3] Although it has been shown that IDO with fully general exceptions (i.e. cancellation links) is exponentially hard [Selman 89]; we've demonstrated that IDO with multiple inheritance exceptions can be computed in polynomial time and is efficiently parallelizable.

the presence of exception handling, although slightly less expressive than Loom due to the lack of extensive numerical capabilities. A full description of the language, and more details on past results can be found in http://www.cs.umd.edu/projects/plus/Parka.

One of the key features of Parka-DB is that it has been shown to efficiently handle inferencing on KBs containing millions of assertions.[4] Early work on the Parka system gained most of its efficiency through massive parallelism, however in recent years we've made increasing use of database management techniques to remove the need for parallelism (although still allowing for efficient parallelization). The version we describe in this paper uses DBMS technologies to support inferencing and data management. This makes for many advantages that will be described later on, but primarily focus on providing support for large scale ontologies.

This paper focuses on the use of Parka-DB in support of medical informatics, but it grows out of previous work done in the area of case-based planning (CBP). The first system to use Parka for CBP was Brian Kettler's *CaPER* planner [Kettler 94]. In this system, the parallel version of Parka was used to handle planning based on very large, automatically generated case libraries. The Caper hybrid ontologies, containing both planning knowledge and many exemplers, were the largest frame-based ontologies created to that point – the largest contained more than 100,000 frames 2,000,000 assertions.[5]

Building on the success of *CaPER*, a joint development effort was begun with BBN to use the databased version of Parka to support a logistics employment planner for military operations [Rager 97]. This system requires the use of two large hybrid ontologies – one for representing knowledge of previous plans (containing more than 50,000 frames and 700,000 assertions) and a second for interoperating with a military database of geographical locations (containing more than 175,000 frames and 1,500,000 assertions). The technologies developed for scaling to these sizes, and for loading data from databases, comprise the basis of the medical application, and we describe these in the next section.[6]

2.2 The Parka-DB architecture

To support the demands of the extremely large KBs which are needed in applications such as those above, we discovered that many of the techniques we used to support MIMD-style parallel computing could be replaced by database operations in a serial system. This led to a reimplementation of the system, which we call Parka-DB, in which the Parka system directly uses a relational database management system (RDBMS). The system thus blends the knowledge-based inferencing capabilities of Parka with the standard database capabilities of the RDBMS. In particular, Parka-DB uses the RDBMS as a run time storage medium. Thus, since the RDBMS uses external devices to store data, Parka-DB can manage knowledge bases that are too large to be maintained in internal memory. In fact, Parka-DB can handle

---

[4] We report KB sizes in "assertions," basically the number of links in the semantic network corresponding to the frames, as this accurately reflects the total number of relations between items in the KB and corresponds directly to the ``concepts'' in a description logic representation.

[5] The planning case bases used in CaPER are available for downloading on the world wide web at the web page cited earlier.

[6] Parka-DB is also being used to support a number of other applications including natural language processing and machine translation, knowledge discovery in databases, and web agents that use SHOE, an ontological extension to HTML. In addition, a request for US and International patents on these algorithms has been filed. Details and papers on other uses of Parka are available from our web page.

KBs that are as large as available disk space, and can make use of the RDBMS to efficiently manage the I/O between primary and secondary storage. This allows Parka-DB to scale to arbitrarily large KBs while still outperforming other KR systems. Details of this implementation are beyond the scope of this paper, but we briefly outline some of the important features of the system.

2.2.1 KB and DB integration

To maintain efficiency in inferencing, we differentiate two categories of concepts: structural and non-structural assertions. Structural assertions consist of the *isa, instance, instanceof* and *subcat* relations that encode the class/subclass ontology in a PARKA KB. Non-structural assertions are all other concepts in the KB. This differentiation is important from a practical viewpoint – the structural assertions are used in property inheritance. In particular, PARKA-DB must scan the structural assertions for computing inherited properties, and inheritance evaluation is a critical aspect of all queries issued to PARKA-DB. Thus, the structural assertions are kept in a special structure for fast access. In all of the cases we've examined, the number of structural assertions is less (often much less) than the number of non-structural assertions. Therefore, we can usually keep these structural assertions in an internal cache even for our largest KBs.[7]

However, while the structural assertions can be kept in memory, the knowledge bases we are using require more memory than is available on all but the largest current computer systems (particularly supercomputers). They certainly don't fit in the amount of internal memory expected to be found in "affordable" computer systems any time in the near future. Furthermore, we find that it is very unlikely that all of the data contained in the set of non-structural assertions will have to be processed for a given query (unlike the structural assertions, which are often all scanned). Therefore, the non-structural assertions are not required to be kept in primary memory. For these reasons, Parka-DB stores non-structural assertions in specialized tables within the database. This allows the database system to handle the loading of only that subset of non-structural assertions required to evaluate a given query, a process which DBMS systems manage efficiently. Thus, to summarize, by dividing data into two categories, Parka-DB can access the highly utilized structural assertions quickly, and rely on the database system to manage the much larger and less frequently accessed individual items of non-structural data efficiently.

To facilitate the efficient moving of data between the inferencing algorithms and the database, we have had to slightly modify the database system. This will be described in the next section, but first we wish to briefly clarify the interaction between the two components. Parka, as a KR system, is primarily concerned with answering queries. In the process of evaluating these queries, Parka issues two main requests to the database. Figure 1 illustrates the interaction between the modules. One request is a command to project data from a non-structural assertion table. The other is a request to perform a join over two tables.

In this manner, Parka-DB utilizes the processing power of the database for those services that a RDBMS can perform more efficiently, mainly join processing. In fact, Parka-DB's query algorithms have been designed such that many of the computations can be directly executed in the form of a join over two relational tables. Thus, the KR component of Parka-DB relies on the database not only for storage and retrieval services, but also for the intermediate

[7] We have recently implemented a version of Parka that pages the structural links when the ISA hierarchy is too large to fit in memory. We have only needed this for testing of our algorithms on random networks containing more than ten million assertions in a structural hierarchy 100 levels deep, and have not needed to use it on even our largest real applications.

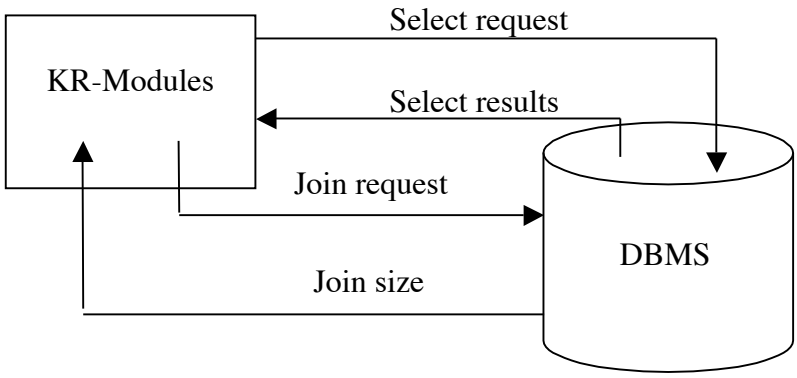


Figure 1: Communication between KRs and DBMS

processing of complex queries. In this manner, Parka-DB can also take direct advantage of database optimizations (for example, the use of parallelism for joins in a parallel DBMS).

2.2.2 PARKA-REL

Parka-DB is implemented on top of an RDBMS system we have created at the University of Maryland. This system, Parka-REL, offers the standard DDL interface for creating databases. Creating relational tables is straightforward and can be achieved through direct function calls from a C program or through a parser. To offer efficient access to user data, the system provides users the opportunity to create indexes on attributes of relational tables. Once the tables have been defined and created, users can seamlessly insert and delete records. Parka-REL provides storage and retrieval functionality, and all of the standard database operations required in the framework of Parka-DB. The subsystems communicate directly through the RDBMS's library of C function calls, rather than through SQL (although a limited SQL interface is provided for user access to the DB). The RDBMS, though limited in functionality, is very useful and supports Parka-DB adequately. We believe the limitations of Parka-REL are more than outweighed by the fact that Parka-REL can be made publicly available and thus Parka-DB distributed without the need for users to buy an expensive DBMS system. In addition, getting the maximum efficiency out of the database has required changing many of the internals (such as optimization and caching functions), and thus the algorithms in Parka-REL have been modified to support Parka-DB as efficiently as possible.

2.2.3 Data Storage

One of the most important aspects of our implementation is that although Parka-DB is in all ways a frame-based KR system, it's knowledge base is actually stored as relational tables to take full advantage of Parka-REL's processing services. In particular, each assertion type is associated with a relational table in Parka-REL. Frames in Parka-DB thus have their information distributed amongst multiple tables.

This contrasts with other attempts to use database technology for supporting knowledge bases and also with object-oriented databases. These other systems store frames in a much more centralized manner -- they attempt to gather as much information about a frame as possible when loading data from external storage. Parka-DB, however, only loads into memory that information which is needed for processing a query. In fact, we believe that one reason

Parka-DB is more efficient than these other systems is that its inferencing algorithms have been designed to take full advantage of this distributed relational (frame) data [Stoffel 97b].

When evaluating complex queries on large knowledge bases, the sizes of intermediate results can be enormous. Computing the results of such queries using only internal memory can place a big demand on system resources, especially if the data to be processed is also stored in internal memory (as in most KR systems). Parka-DB overcomes this problem by distributing the workload such that the KR component prepares relational tables for the RDBMS component to process using external storage as it needs. The RDBMS unit can consume available disk space while processing intermediate results thus reducing the demand for valuable internal system resources.

3 The Medical Application of Ontologies

Parka-DB has been used as an underlying technology in the support of a joint effort between computer scientists at the University of Maryland and clinicians at Johns Hopkins Hospital (JHH) which focuses on using high performance computing technology in support of medical applications. One focus of this effort is on providing tools that support medical data warehousing, and our particular project focuses on providing support to micro-biologists and infectious disease specialists.

A problem we have encountered is the difficulty clinicians and researchers face in formulating database queries that capture the kinds of questions they wish to pose: the databases are not based upon the terminology most familiar to the clinicians, nor do they support the semantic relations the doctors use in understanding their medical taxonomies. As a means for overcoming this sort of difficulty, we are looking at how to efficiently integrate "semantic" knowledge, stored in the form of Parka-DB ontologies with low-level data to allow the efficient accessing of large databases. Parka's ontology management capabilities are used to provide medical specialists with the ability to express complex queries without becoming experts on the underlying data model. The ontology helps us to facilitate complex data access, and to support high-level querying for users untrained in the details of the underlying database forms.

3.1 Motivating Examples

As stated above, our examples are motivated by a group of applications that involve the analysis of information in a medical data warehouse we are constructing at Johns Hopkins Hospital. The data warehouse includes in-patient and outpatient data from the microbiology, blood bank, clinical chemistry and hematology laboratories, along with pharmacy data and some clinical data. The microbiology data is obtained by carrying out a large and heterogeneous collection of identification techniques, which are generally carried out in a stepwise manner. One initially discovers that the organism to be identified belongs to one of several very large classes of bacteria, fungi or viruses. Protocols are then followed to produce a more precise identification. (The scope of this paper precludes a detailed discussion of microbiological taxonomy, see [Bergey 84] for the medical details).

A large and heterogeneous set of criteria have evolved for classifying microorganisms. A bacterial species can be defined by:

1. structural attributes of size, shape, Gram stain reaction and macroscopic growth,
2. physiologic traits relative to oxygen, temperature, pH,

3. biochemical and nutritional traits, biochemical information on cell composition and metabolites, and
4. DNA sequence information.

Description and identification of potentially infectious microorganisms are generally carried out to guide clinicians in their choice of antimicrobial therapy. Microbiological identification requires a process of iterative refinement; there is not a single battery of tests that can be applied to all specimens to obtain a precise identification.

The set of protocols required to precisely identify an organism can be represented as a directed acyclic graph. Most organisms are not precisely identified as the iterative process of organism identification continues only as long as clinically useful information is obtained. A highly imprecise identification is permissible as long as the organisms identified are not likely to be causing harm to a patient. Identical organisms may be identified to different degrees of precision as clinical requirements may vary from patient to patient. Moreover, the set of protocols used to identify organisms are modified from time to time, and identification protocols may differ somewhat from laboratory to laboratory.

The complex taxonomy used to describe microorganisms and the non-uniform degree of microorganism identification create difficulties for those who wish to pose queries to a clinical data warehouse. The clinical context associated with these examples is an ongoing effort to characterize microorganism antibiotic susceptibility. This issue is of continuing concern because of the continuing emergence of antibiotic resistant microorganisms. Thus, a typical desire might be to *"Find all patients on the oncology service whose cultures grew gram negative aerobe rods that were resistant to Piperacillin and Trimethoprim/Sulfamethoxazole."*

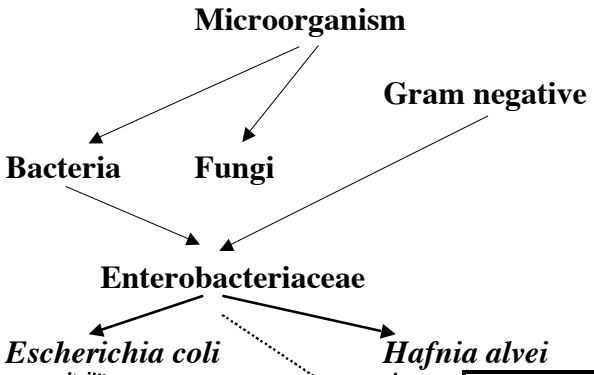
For a clinician or medical researcher, it is difficult to produce a query like this on the fly. The database terms used to express things such as "oncology service" or "gram negative aerobe rods" may require many conjuncts of low level terms. Using Parka, we are able to associate these terms with items in the database, and then to load these into a Parka-DB knowledge base. Thus, we essentially find all the instances corresponding to terms in the ontology, load these into the KB, and then perform Parka-DB queries as a back-end for a GUI that is usable by the doctors. In the remainder of this section we describe this process in more detail, and show how the ontology helps the medical personnel to issue queries such as the one above.

3.2 Using Ontologies for Indexing

In order to use the ontologies for database access, we have to establish links between the data in the data warehouse and the concepts in the ontology. For each node in the ontology we create a set of links to all tuples[8] which have a value for one of the attributes described by the ontological term. An example is given in Figure 2. The solid links are ontological links and the dashed links are indexing links.

Thus for example the links from the concept "Enterobacteriaceaes" to tuple 1 and 2 indicates that there is a relation between these tuples and this concept in the ontology (Enterococcus and Streptococcus belong to the Family of Enterobacteriacea). In order to automate the

[8] Throughout this paper we will act as if the data is stored in a universal relation. This is not a necessary condition, but simplifies the discussion of the basic ideas



Patient	Microorganism	Age
AA	*Escherichia coli*	34
BB	*Enterobacteriaceae*	52
CC	*Hafnia alvei*	19
DD	*Escherichia coli*	26

Figure 2: Ontology with links into the database.

indexing procedure we have developed a tool, the PDBCT (Parka - Database Connectivity Tool), which forms the basis for the medical application[9].

3.3 PARKA-Database Connectivity Tool (PDBCT)

The PDBCT is a generic mechanism, which allows the linking of Parka-DB to virtually any relational database. Applications that use Parka-DB and the PDBCT can manage very large ontologies in the Parka part of the system while, through the PDBCT, they can interoperate in a natural way with data in a database. This information is then brought in as instances to the ontology, essentially creating a hybrid ontology on demand. This ontology can then be queried or browsed using a generic Parka-DB front end (available in both C and Java) or using a specialized front end designed for the needs of the specific users.

As a slightly less technical example of this than the one presented earlier, consider a researcher who wishes to do research into the effects of smoking on pregnant teenagers. In particular, the researcher wishes to explore how many of the patients in a large hospital database are teenage (i.e. some age field is between 13 and 19) and who have a diagnosis that is a subclass of "oncology." Using the PDBCT, the researcher browses the pure ontology, and finds the necessary terms (oncology and teenager). This is then built, by the PDBCT, into a SQL query that goes out to the database and loads all the fields concernig patients with these two specialized properties. This is loaded into Parka-DB, and thus the researcher now can ask queries or browse in the hybrid ontology that contains all the "pure" medical ontology terms coupled with all the instances that are germane to the research.

[9] The PDBCT is not strictly a medical tool, it can actually be used for any application in which we wish to load data into a knowledge base. In this paper, however, we only discuss the medical example.

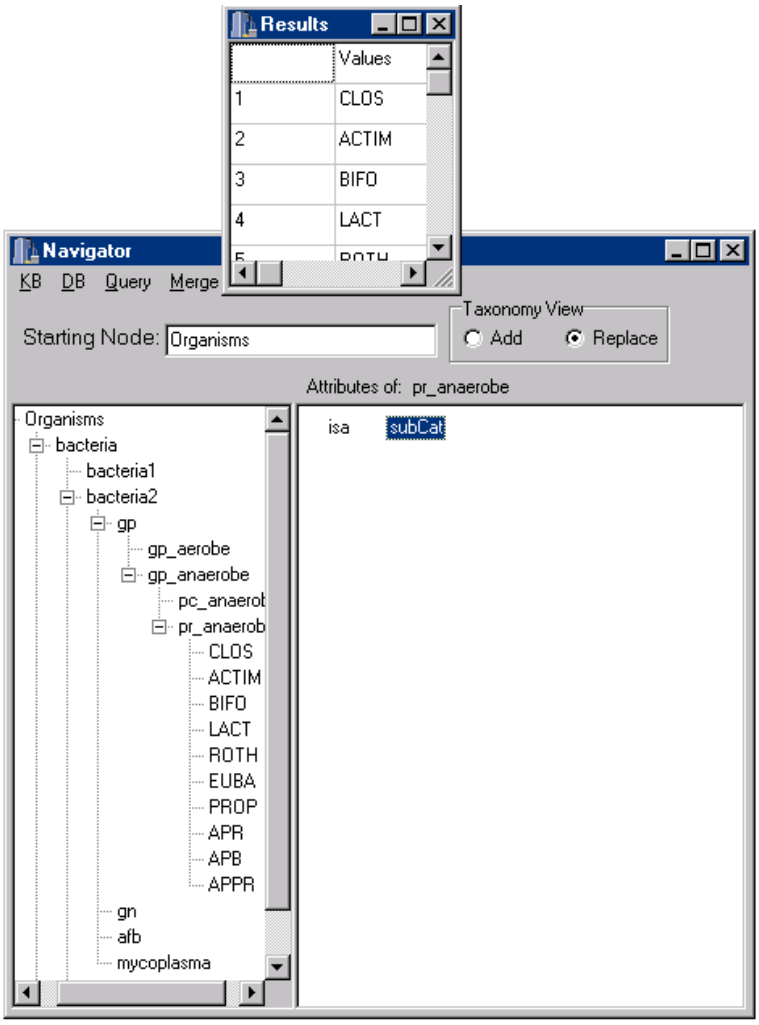


Figure 3: PDBCT graphical interface: Left hand side – hierarchical display of the structural (Isa) frames in the ontology; Right hand side – list of attributes and/or properties of a selected frame; Top – the chosen attributes displayed (in this case, the list of subcategories of *pr_anaerobe*) for further browsing or editing.

3.4 PDBCT Architecture

Figure 3 shows the graphical interface used by doctors for interacting with Parka-Db and the PDBCT. The left part of the screen represents the taxonomical part of the knowledge base (essentially the ISA hierarchy). (The taxonomy is presented as a tree although the underlying structure is really a DAG). The right part of the system lists the attributes (slots) of the frame selected in the left part of the system. By double clicking on a frame in the left window the taxonomy can be expanded, i.e. the successors of the selected node are displayed in the tree view. By clicking on a attribute listed in the right part of the window a grid will popup and display the values for the frame selected in the left part of the window and the attribute selected in the right part (right upper part of Figure 3). All these operations are standard knowledge base operations and are realized in the Parka part of the system.

The PDBCT offers a set of "virtual relations" that are directly visible from KR part of the Parka system. To access data provided by the database system the relevant data has to be loaded into these virtual relations (i.e. instances must be loaded from the database to the

knowledge base). This is realized through a query interface in which SQL queries can be formulated, passed to the database, and the retrieved data stored in the virtual relations. These virtual relations can be merged with the pure ontological information, producing the new hybrid ontology.

Every frame possesses a set of rules that describe under which conditions a database tuple is accepted as an instance of the frame. In most application very simple rule systems are sufficient to define the conditions under which a tuple is accepted. For our medical databases, it was sufficient to define a list of attribute value pairs that have to be found in the tuple in order to insert them as instances for a given concept. E.g. the instances of the concept "teenager" could be described as all tuples where the field "Age" is greater than 12 and less than 20. Data is merged from these virtual relations into the ontology simply by checking these rules, i.e. if the rules for a concept in the ontology fire, then the identification of the current tuple gets added as an instance of that concept.

As an example of the power of using the ontology to help in accessing the appropriate data, we show how the doctor accesses information using the Parka-DB system vs. how it would be done in the raw database system. We will use the medical example presented earlier: *: "Find all patients on the oncology service whose cultures grew gram negative aerobe rods that were resistant to Piperacillin and Trimethoprim/Sulfamethoxazole"*. We note that in the example we'll show, all patient data is artificial – actual hospital data is restricted due to US Federal and Maryland privacy laws.

In posing our query, the user has to select the restricting nodes in the right part of the window (see Figure 4), i.e. selecting the nodes *"gram negative", "rods","aerobe", "resistant-Piperacillin", "resistant-Trimethoprim/ Sulfamethoxazole"* and *"oncology service"*. Afterwards he has to select the *"Get all instances"* from the popup menu. By doing so the user is selecting all identifications for tuples in the database that meet the selected criteria.

A grid with all tuple identifications shows up (Figure 4 top-left). Following this, the user double clicks on the Grid to select the database. (The version presented in this paper is the PC based version of PDBCT which uses MS-ACCESS as its database system.) The tuple Ids that were found in the Grid are now selected and presented with all their attributes in form of database relation (right part of Figure 4):

The representation for this query in the PARKA internal language varies with the implementation of the KB that represents the medical knowledge. Our KB contains a concept that defines *"gram negative"* and another that defines *"rods"*. Under this condition it is fairly easy to formulate a PARKA-query that retrieves the tuple identifications. During the merge process (see above) we inserted the tuple identifications into the KB by assigning them as instances to related concepts. E.g. if a database tuple specifies that the patient was infected by a gram negative bacterium the identification of this tuple gets added as an instance of the concept *" gram negative"* to the database. To find all tuple identifications which are answers to our query, we have to find all tuples that are instances to all of the 6 concepts that define the constraints for the query. The Parka-DB form of this query is:

```
(instance gram-negative ?x)(instance rods ?x)(instance
aerobe ?x)(instance resistant-Piperacillin ?x)(instance
resistant-Trimethoprim/Sulfamethoxazole ?x)(instance on-
cology-service ?x)
```

which is a conjunctive query that returns all items in the ontology with which "?x" unifies. (In this case the query is basically a recognition query, in other instances it may be more complex containing multiple variables. For examples of significantly more complex medical queries [Stoffel 97a].)

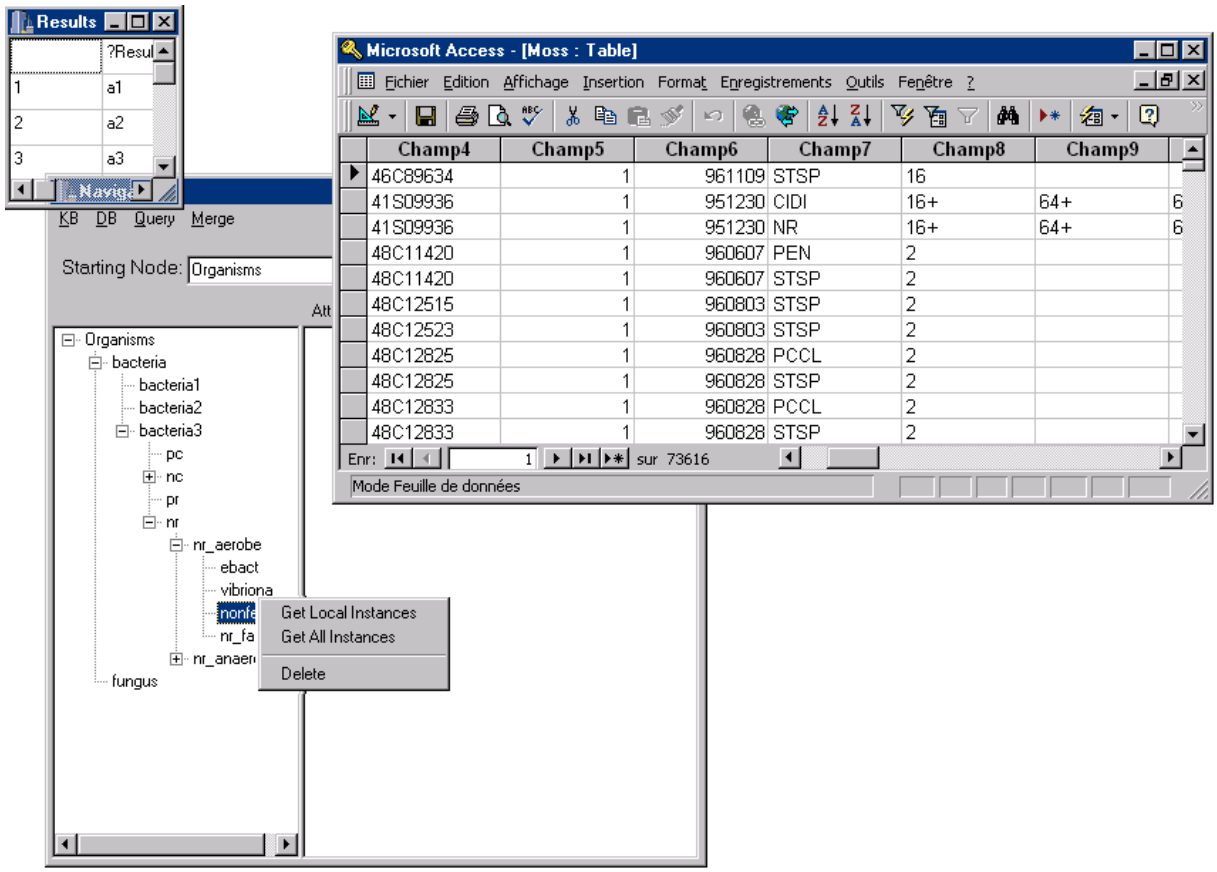


Figure 4: Querying a DB from the PDBCT graphical interface: Top left -- A grid with all tuple identifications; Right hand side -- Tuple Ids are selected and presented in the form of a database view .

To really see the advantage of using the ontology in database access, consider what the same user would need to do issue the SQL query directly. The user has to create a complex SQL expression that directly lists all the base terms in the ontology (i.e. this would be like listing the fifty individual states, rather than asserting the single term "state.") Thus, the user would have to express all the oncology wards, all the organisms that have the correct characteristics, the various condition under which the correct resistances are found, and the actual values for the various fields that specify these things. Suppressing some of the details, the query would look like:

```
SELECT * FROM Moss
     WHERE ((
      Location IN (`onc-ward-1',,'onc-ward-2','onc-clinic-1','onc-clinic-2', (9 elements sup-
      pressed))))
      AND ((
      Organism IN ('gram negative aerobe','gram negative rod aerobe', 'e coli',… (298 elements
      suppressed))))
      AND ((((
      Organism IN ('Enterobacteriaceae','Escherichia species',' Escherichia coli','Escherichia
      coli O157:H7'…(114 elements suppressed)
      OR (
      Organism IN('nonfermenter','Pseudomonas species','Pseudomonas aeruginosa', …(63 elements
      suppressed)
      AND ((
      TrimethSulfMIC IN ('ResistantTrimethoprim/SulfamethoxazoleEnterobacteriaceae', '2+/38+'
      ))))
      OR (((
      Organism IN ('Pseudomonas species','Pseudomonas aeruginosa','Pseudomonas alcali-
      genes','Pseudomonas fluorescens',  , (7 elements suppressed)
```

```
AND ((
TrimethSulfMIC IN ('ResistantTrimethoprim/SulfamethoxazolePsuedomonas ','2+/38+')))))
AND
((((
Organism IN ('Enterobacteriaceae','Escherichia species',' Escherichia coli',' Es-
cherichia coli O157:H7'...(114 elements suppressed)
AND ((
PipericillinMIC IN ('ResistantPiperacillinEnterobacteriaceae','64+','64.1+'))))
OR ( ((
Organism IN (''Pseudomonas species','Pseudomonas aeruginosa','Pseudomonas alcali-
genes','Pseudomonas fluorescens',    (7 elements suppressed))
AND (
PipericillinMIC IN (' ResistantPiperacillinEnterobacteriaceae','64+','64.1+'))))));
```

Note that this is not only very difficult to formulate, but also that a lot of medical and hospital internal knowledge is needed to build this query. Thus, previously clinicians had to interact with a database specialist to formulate the query, and then use database tools to examine the results. In short, what was previously a batch process needing human intervention by a database specialist, now becomes an interactive process using the natural ontological terms used by the medical, as opposed to database, specialists.

Once the information is returned from the database, the user has a choice. The information can be seen as a database table and/or it can be loaded into Parka-DB as a hybrid ontology (one with many instances). In the latter case, the ontologies created can be browsed or queried using the tools Parka provides [Stoffel 97a]. Thus using the Parka-based system, the user uses a single interface to formulate the query, access the data, and browse the results.

The scaling provided by Parka-DB is crucial to this latter – the hybrid ontologies can get very large. For the example we've been discussing, the "pure" microbiological ontology contains only about 1200 assertions. The database, however, contains about 75,000 records each with about forty properties. Thus, the hybrid knowledge base created may contain as many as 3,000,000 assertions – requiring the capabilities of Parka-DB to browse and query.

As if this wasn't enough, we have begun working with much larger pure ontologies, such as the UMLS system, which contains over one million assertions (as discussed in [Stoffel 97b]). One of our goals is to hook this much more general ontology up to the hospital's financial database, which contains information on every charge, by every patient, for every bed in this large (1,000+ bed) hospital. A year's worth of this data, for example, could require as many as a hundred million assertions – just for this one hospital! For a large HMO or medical insurance carrier, the situation could be orders of magnitude worse. The scaling provided by Parka's parallel algorithms are a partial solution to this problem, but we are also examining new mechanisms to distribute the data and knowledge bases so as to handle the enormous knowledge bases that future applications will require.

4 Conclusions

As stated in the introduction, modern information technology systems place more demands on ontology management and support tools than have many previous AI-based applications. The need to support very large ontologies, and to integrate ontological and database information, has been a driving force in the development of the Parka-DB system. In this paper, we have described one application of the use of our system in support of a medical application. The ability to present users with ontological data that matches their knowledge of the domain, and thence to use that knowledge in finding and interacting with the actual hospital data in real time, presents the users with new capabilities not found in either traditional database or AI tools. The integration of the two is a powerful technique, and we believe applications like

this one barely scratch the surface of the capabilities that modern advances in both ontology management and database interoperability provide.

5 Acknowledgements

This research was supported in part by grants from ONR (N00014-J-91-1451), ARPA (N00014-94-1090, DABT-95-C0037, F30602-93-C-0039), and the ARL (DAAH049610297). Dr. Hendler is also affiliated with the UM Institute for Systems Research (NSF Grant NSF EEC 94-02384). We particularly thank Merwyn Taylor who was one of the key implementors of Parka-DB and our coauthor on several earlier papers about the system.

6 References

[Rager 97] D. Rager, J. Hendler, and A. Mulvehill, ForMAT and Parka: A technology integration experiment and beyond, Proc. *Intl Conference on Case-Based Reasoning* ,Providence, RI, 1997.

[Kettler 94] B. Kettler, W. Andersen, M. Evett and H. Hendler, Massively Parallel Support for Case-based Planning. *IEEE Expert* , February 1994.

[Lenat 91] Douglas B. Lenat and R.V. Guha. Building Large Knowledge-Based Systems. Addison-Wesley, 1991.

[Knight 94] K. Knight and S. Luk. Building a Large Knowledge Base for Machine Translation. In *Proceedings of the 12th National Conference of the American Association for Artificial Intelligence*, AAAI Press, MIT Press, 1994.

[Chaudri 98] V. Chaudri, A. Farquhar, R. Fikes, P. Karp, and J. Rice, OKBC: A Programatic Foundation for Knowledge Base Interoperability, *Proceedings of the 15th National Conference of the AAAI (AAAI-98),* AAAI Press, 1998.

[Stoffel 97a] Kilian Stoffel, Joel Saltz, Jim Hendler, Jim Dick, William Merz and Robert Miller. Semantic Indexing for Complex Patient Grouping. In *Proceedings of the Annual Conference of the American Medical Informatics Association*, 1997.

[Stoffel 97b] K. Stoffel, M. Taylor, and J. Hendler, Efficient Management of Very Large Ontologies, *Proceedings of the 14th National Conference of the AAAI (AAAI-97),* AAAI Press, 1997.

[Miller 96] George A. Miller. HUMAN LANGUAGE TECHNOLOGY. In *Technical Report*, Psychology Department, Green Hall, Princeton University, 1996.

[UMLS 94] UMLS. "Unified Medical Language System". National Library of Medicine, 1994.

[Code 93] Roger Code. Systemized Nomenclature of Human and Veterinary Medicine: SNOMED. College of American Pathologists, American Veterinary Medical Association, 1993.

[Gruber 90] T. Gruber. The Development of Large, Shared Knowledge-Bases: Collaborative Activities at Stanford, In *Technical Report*, Knowledge Systems Laboratory, 1990.

[Gruber 92] T. R. Gruber . Ontolingua: A Mechanism to Support Portable Ontologies. In *Technical Report*, Knowledge Systems Laboratory, 1992.

[Sowa 84] John F. Sowa. Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, 1984.

[MacGregor 94] Robert M. MacGregor. A Description Classifier for the Predicate Calculus. In *Proceedings of the 12th National Conference of the American Association for Artificial Intelligence*, AAAI Press, MIT Press, 1994.

[Horty 90] J.F. Horty, R.H. Thomason and D.S. Touretzky, A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks, *Artificial Intelligence*, 42(2-3), 1990.

[Selman 89] B. Selman and H. Levesque, The Tractability of Path-Based Inheritance, *Proceedings of IJCAI-89*, Morgan-Kaufmann, 1989.

[Borgida 94] A. Borgida and P.F. Patel-Schneider. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. In *Journal of Artificial Intelligence Research*, Volume 1, pp. 277-308, 1994.

[Shapiro 92] Stuart C. Shapiro and William J. Rapaport. The SNePS Family. In *Computers \& Mathematics with Applications*, pp. 243-275, 1992.

[Baader 94] Franz Baader and Bernhard Hollunder and Bernhard Nebel and Hans-Jürgen Profitlich and Enrico Franconi. An Empirical Analysis of Optimization Techniques for Terminological Representation Systems or "Making. KRIS get a move on". In *Applied Intelligence*, Volume 4, Number 2, pp. 109-132, May, 1994.

[Haase 96] Kenneth Haase. FramerD: Representing knowledge in the large. In *IBM Systems Journal*, 1996.

[Lee 96] Eunice Lee and James Geller. Parallel Transitive Reasoning in Mixed Relational Hierarchies. In *Principles of Knowledge Representation and Reasoning*, Luigia Carlucci Aiello and Jon Doyle and Stuart Shapiro, 1996.

[Evett 94] W. Andersen and M. Evett and J. Hendler and B. Kettler. Massively Parallel Matching of Knowledge Structures. In *AAAI/MIT Press*, H. Kitano and J. Hendler, Series Massively Parallel Artificial Intelligence, 1994.

[Karp 95] Peter D. Karp and Suzanne M. Paley. Knowledge Representation in the Large. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, pp.751-758, 1995.

[Shastri 93] L. Shastri and V. Ajjanagadde. From simple associations to systematic reasoning. In *Behavioral and Brain Sciences*, Volume 16, Number 3, pp. 417-494, 1993.

[Bergey 84] I. Bergey. Systemactic Bacteriology. Williams and Wilkins, 1984.